MSc Artificial Intelligence
Master Thesis

Visual Feature Detection in RTAB-Map with the Spot Robot: Applicability and Performance Analysis

by
Emily Sae Mes
11737220

July 26, 2023

48 EC
November 2022 - July 2023

Supervisor:
Prof. Dr. Takayuki Nagai
Examiner:
Dr. Arnoud Visser

Second reader:
Dr. Shaodi You

Universiteit van Amsterdam

Contents

1 Introduction **2**

2 Problem definition **3**

3 Theoretical background **5**
3.1 Navigation . . . 5
3.2 Simultaneous Localization and Mapping (SLAM) . . . 6
3.2.1 Problem definition . . . 6
3.2.2 SLAM paradigms . . . 7
3.3 SLAM implementations . . . 8
3.3.1 Filter-based SLAM . . . 8
3.3.2 Graph-based SLAM . . . 8
3.3.3 Visual SLAM . . . 8
3.4 VisualSLAM methods . . . 9

4 Related work **10**
4.1 Feature Detection . . . 10
4.2 Binary features . . . 10
4.2.1 Comparison . . . 11
4.3 Floating-point features . . . 12
4.3.1 Comparison . . . 14
4.4 Deep learning-based features . . . 14
4.4.1 Comparison . . . 16
4.5 Discussion . . . 17

5 Method **18**
5.1 Robot Spot . . . 18
5.2 Hardware . . . 19
5.3 Software . . . 20
5.3.1 ROS . . . 20
5.3.2 RTABMap . . . 20
5.4 Implementation . . . 23
5.4.1 RTABMap addition: R2D2 . . . 23
5.4.2 Evaluation . . . 24
5.5 Summary . . . 26

6 Experiments **27**
6.1 Perspective . . . 27
6.2 Validation . . . 28
6.3 Extension . . . 29

7 Results **30**
7.1 Perspective 30
7.2 Validation 31
7.2.1 Exploration 31
7.2.2 Qualitative results 32
7.2.3 Quantitative results 34
7.3 Extension 35
7.4 Conclusion 36

8 Conclusions **37**
8.1 Discussion 37
8.2 Future work 37
8.3 Conclusion 38

Abstract

This thesis investigates the applicability and performance of feature detection in a visual SLAM approach, namely RTAB-Map, by conducting a comparative analysis of existing visual features in terms of re-localization and application and the integration of an additional visual feature using the robot Spot. This study aims to gain a deeper understanding of the optimal performance of visual SLAM under varying conditions. This study finds that the optimal choice of a feature detector depends on the specific task at hand and the data distortions, leading to diverse outcomes, notable with the SuperPoint feature. Moreover, the integration of the R2D2 feature detection may have revealed a concern about using deep-learning based visual features in combination with visual SLAM. Additionally, an evaluation approach based on the fiducial principle is proposed for map assessment in the absence of ground truth data. In conclusion, this thesis provides valuable insights into visual SLAM feature detection, and offers directions for future research to enhance the methodology's robustness and overall performance.

CHAPTER 1

Introduction

Japan has been facing a rapid shift in the demographic structure since 2010 [49], which is demonstrated by the declining birthrate and aging population [58]. This also results in a decreasing labor force; Whereas labor force was around 79 million people in 2013, it is expected to decline by 9 million people in 2025 [66]. A domestic help robot could provide a solution for both the demanding elderly care and decreasing labor force, for example by being able to send a robot on a mission to get products from a store and deliver it back to you. Since convenience stores are structured and its services are standardized, it is an ideal environment for this study. Furthermore, due to Japan having around 50.000 convenience stores [66], the impact could also be substantial. Robotics in convenience stores has been studied scarcely, however, its research has been accelerated by the introduction of the Future Convenience Store Contest (FCSC) in 2017 [66], which focused on tasks such as restocking, trash disposal, customer interaction and restroom cleaning. The robot Spot could be used as test object for this kind of application, since it can be used both indoors and outdoors, has the ability to perform actions, such as opening doors and picking up objects, and is able to descend and climb stairs. Furthermore, it has the required sensors and cameras to perform the tasks. Therefore, for this research Spot will be used to provide a understanding and lay the groundwork to achieve this envisioned scenario.

CHAPTER 2

Problem definition

Since the release of the robot Spot, it has been used in a variety of applications or research, such as exploring extreme environments [4] or joining search-and-rescue missions of the military or police departments [45, 19, 70]. The fact that Spot has mostly been used in those kinds of applications comes as no surprise, since Spot started as a Defense Advanced Research Projects Agency (DARPA)-funded project in as early as 2003 [70]. This may have led to perceptions of dystopian danger. Nonetheless, Boston Dynamics has also shown the entertaining side of Spot while showing off its dance moves and its human-friendly domestic side when fetching a drink for its owner at home in various published videos [1] [2] [3]. Spot is very suitable for comprehensive tasks, due to its versatility and agility [23].

Service robots in retail environments have been experimented with for a variety of tasks, such as showing product locations or carrying products for customers [21]. Additionally, recently a robot restocking the shelves has been rolled out to hundreds of Japanese convenience stores [4]. These service robots in retail environments aim to improve overall customer experience, in which it succeeded since [21] reported that 92% of the customers would use a robot again as a shopping companion. However, in those experiments, full prior knowledge of the environments was required, meaning that the robot can not operate autonomously in an environment that is not manually mapped, and can not adopt to changes in that environment. Incorporating SLAM could solve these challenges. In [13], a robot architecture for the autonomous search and localization of products in unknown dynamic grocery stores is proposed by using the contextSLAM framework. The contextSLAM framework achieves this by obtaining the context via deep learning-based Optical Character Recognition (OCR), and merging this context with the laser range measurements.

[1] https://www.youtube.com/watch?v=kHBcVlqpvZ8
[2] https://www.youtube.com/watch?v=fn3KWM1kuAw
[3] https://www.youtube.com/watch?v=tf7IEVTDjng
[4] https://blogs.nvidia.com/blog/2022/08/10/telexistence-convenience-store-robotics/

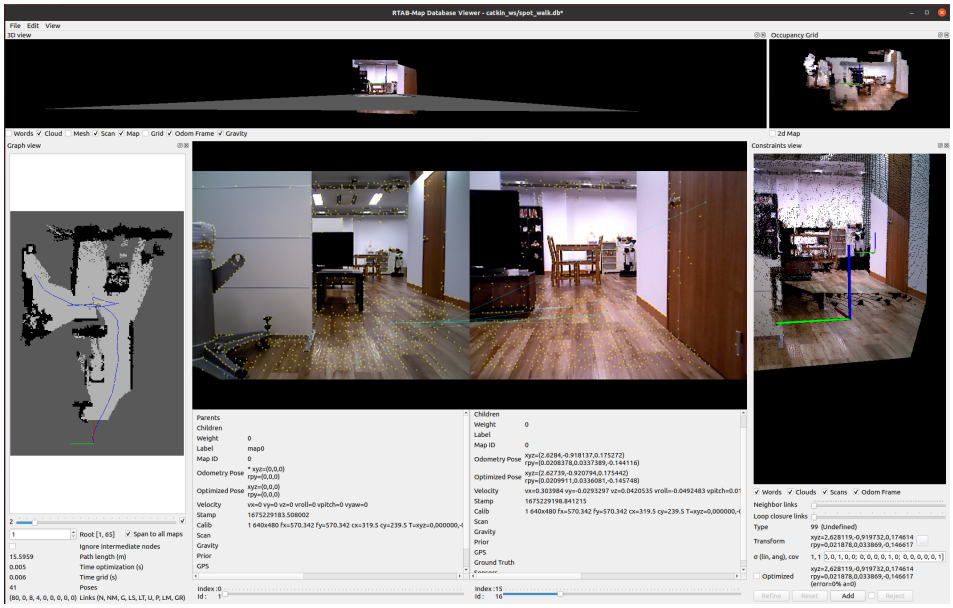


Figure 2.1: Situation of the robot Spot mapping and localizing itself in the Experiment Room of the Robot Learning Group.

In this study, the objective will be for Spot to localize and map itself in an unknown environment. In the end, this could be extended to and have an application in Spot retrieving an item from the convenience store. Specifically, the research will take place in the Experiment Room of the Robot Learning Group of the Osaka University, where various visual features incorporated in SLAM methods will be compared. A situation where the robot Spot is mapping and localizing itself in this environment is shown in Figure 2.1, which shows the construction of the map on the left side by finding keypoints and matches between frames as shown in the middle. While it is also interesting to let the Spot robot walk freely through the Kisokougaku building of the Toyonaka campus where the Experiment Room is located, this had to be restricted too much, as for example navigation with an elevator is a research project on its own [65, 69]. Therefore, the main focus will be on the navigation in the Experiment Room. This gives the following research questions:

RQ1. What are the currently available feature detectors for visual SLAM methods, and how do they compare to each other?

RQ2. Is it possible to integrate a new feature detector, namely R2D2, into RTAB-Map, and how does its performance compare to the existing feature detectors?

RQ3. How can map assessment be performed of a map produced by visual SLAM when the ground truth is not available?

CHAPTER 3

Theoretical background

3.1 Navigation

Navigation is commonly regarded as the most challenging competence that is required of an autonomous mobile robot [59]. Robot navigation consists of multiple parts. First of all, the robot has to interpret its sensory data in order to gain relevant information. Secondly, the robot has to localize itself in the current environment. Thirdly, the robot has to know what to do to achieve its given objective. Lastly, the robot has to direct itself to do what it wants itself to do. These four building blocks – *perception, localization, cognition and motion* – are closely intertwined: however good the processing of information is, if the sensory information is not correctly obtained, the robot will have trouble localizing itself in the environment. Alternatively, when the robot is successful in all four of those modules, it will succeed in navigating itself safely through an environment. The process of robot navigation can be seen in Figure 3.1. The operations of this study are identical to this process. When the Spot robot starts its operation, it must gain information about its current environment (perception) to map and localize itself (localization), whereafter it can start planning where it wants to go (cognition). Lastly, this planned path could be executed and it could complete its given task of for example retrieving an item (motion).

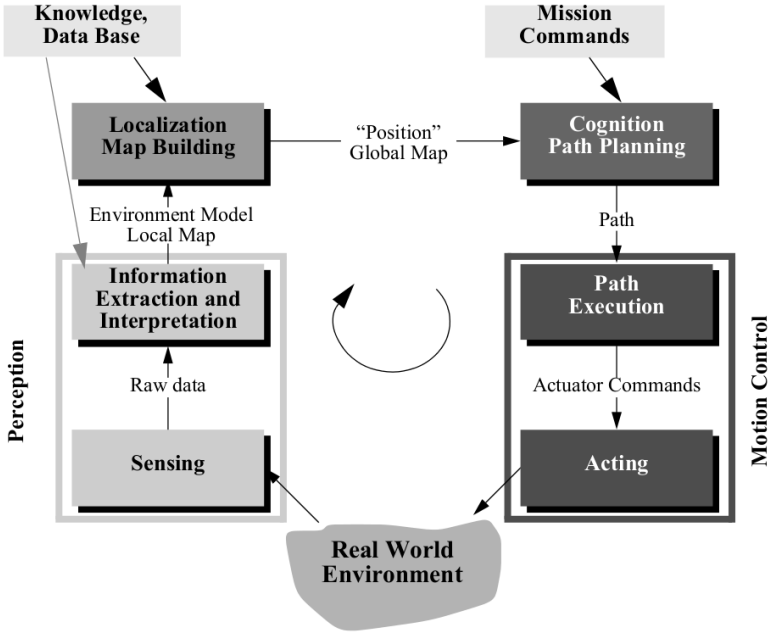


Figure 3.1: An abstract control scheme for mobile robot systems. Courtesy from [59].

3.2 Simultaneous Localization and Mapping (SLAM)

Navigating through an environment requires the knowledge of the current location and the existence of a map of the environment. This map is classically built by hand, which can be hard, costly, and very time-consuming [59]. Localization is the process of using the sensor data to update the current position of a robot in an environment. So navigation is actually a chicken-and-egg problem [24]. The current position of the robot (obtained by localization) is required to build a map of the environment successfully, but the map is essential when performing localization. This is one of the main reasons for 'autonomous map building' [68], which enables a robot to be placed at an unknown location in an unknown environment, where it incrementally builds a consistent map of this environment while simultaneously determining its location on the map [11]. This is called the Simultaneous Localization and Mapping (SLAM) problem [59]. In short, SLAM addresses the problem of a robot navigating in an unknown environment, while seeking to acquire a map of that environment to localize itself on that map. This section will provide the mathematical problem definition and function as a comprehensive introduction to the major paradigms. To put into perspective, in this study the RTABMap visual SLAM algorithm will be used, which is a graph-based SLAM method that is able to process both laser and visual observations.

3.2.1 Problem definition

The SLAM problem can be defined in probabilistic terminology [61]. Let t be a certain time point, and the position of the robot at that time as $\vec{x_t} = (\vec{X_t}, \Theta_t)^T$, where $\vec{X_t} = (x_t, y_t, z_t)^T$ and Θ_t denote the 3D position and the orientation of the robot. The entire trajectory of the robot can then be denoted by $\mathbf{X}_T = (\vec{x}_0, \vec{x}_1, ..., \vec{x}_T)$. Note that T is the time length of the trajectory, and $\vec{x}_0$ is the initial position. The measurements of the odometry between time point t and time point $t-1$ can be denoted by $\vec{u_t}$, meaning that $\mathbf{U}_T = (\vec{u}_0, \vec{u}_1, ..., \vec{u}_T)$ denotes the sequence of relative motions of the trajectory. In theory, this relative odometry sequence in combination with the initial position $\vec{x}_0$ could be enough to determine the entire path. However, odometry readings are prone to be corrupted by noise, which might result in a discrepancy between the actual end pose $\vec{x}_T$ and reconstructed end pose $\vec{x}_T^*$. The actual map of the environment can be denoted as m, which consists locations of possible landmarks or objects in the environment. These locations are determined by the sensor measurements at a certain time step $\vec{z}_t$. This means that the sensor measurements of the robots trajectory can be defined by $\mathbf{Z}_T = (\vec{z}_0, \vec{z}_1, ..., \vec{z}_T)$. Note that here the assumption is made that a sensor measurement is taken at every time step. Now, the SLAM problem is the problem of recovering the robot trajectory $\mathbf{X}_T$ and modeling the map m simultaneously, from the odometry data $\mathbf{U}_T$ and the sensor measurements $\mathbf{Z}_T$. Figure 3.2 shows a graphical model of the SLAM problem.

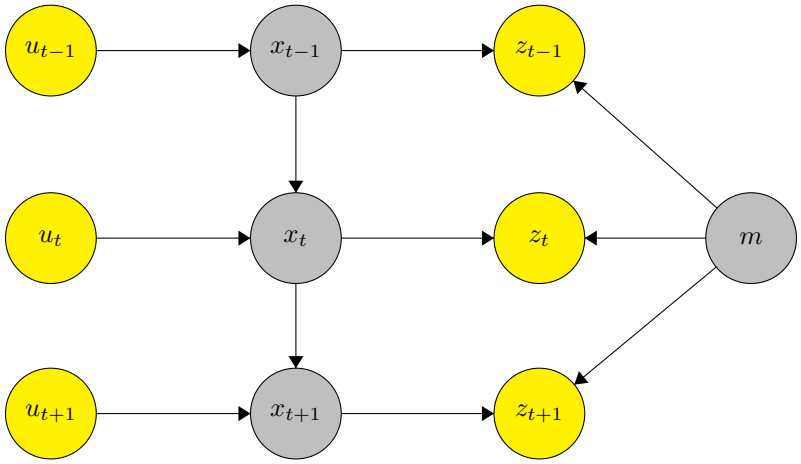


Figure 3.2: A graphical model of the SLAM problem. Note that the arrows indicate a causal relationship, and that the yellow nodes are observable by the robot, whereas the shaded nodes are not. The goal of SLAM is to recover the shaded nodes. Inspired by / courtesy from [61].

The SLAM problem can be divided up into two flavors: the *offline*, also called *full*, and the *online* approach. The offline approach considers the full posterior, thus the measurements of the entire trajectory to estimate the entire trajectory and the map, as follows:

$$p(\mathbf{X}_T, m|\mathbf{Z}_T, \mathbf{U}_T) \tag{3.1}$$

The online approach tries to estimate the current robot pose instead of the entire trajectory, and utilizes only the measurements at one time step, as follows:

$$p(\vec{x}_t, m|\vec{z}_t, \vec{u}_t) \tag{3.2}$$

While both instances seek to maximize the probability of the robot pose(s) and the map, the approach differs, meaning there are different paradigms. These will be discussed in the next subsection.

3.2.2 SLAM paradigms

In this section, the basic paradigms will be discussed, which can be divided up in filter-based and graph-based methods. Note that typically in the filter-based approaches the map is incrementally established whereas the graph-based approaches can typically be categorized as the previously discussed full approach. The **filter-based** paradigm is typically based on the principles of the Extended Kalman Filter (EKF) SLAM and Particle Filter (PF) SLAM. EKF SLAM [2] was historically the first method, but has since then become overshadowed due to its limitations in computation. In EKF SLAM, the pose of the robot and the positions of all features are stored in a state vector and the associated error covariance matrix that represents the uncertainty in these estimates are being updated using the extended Kalman filter as the robot moves through the environment and observes new features. This means that as new features, such as points or planes, are observed, the state vector increases and the covariance matrix is growing quadratically, hence EKF SLAM being computationally expensive. In PF SLAM [46], the pose of the robot and the locations of the landmarks are also iteratively updated as the robot moves through the environment, however, the big difference is that the robot's pose and the landmarks are represented by a set of particles, by which the posterior distribution can be approximated. In theory, the particle filter is capable of approaching the true posterior with an increasing particle size, but since particle filters scale exponentially with the dimension of the true space, its use is limited by the number of dimensions. To summarize, the common feature of the graph-based approach is as follows: the prediction step is model-based on possible paths and maps, which could aggregate uncertainty. The update step then corrects this with the actual observations, which in turn reduces the uncertainty.

The second SLAM paradigm is characterized by the **graph-based** models, which uses non-linear sparse optimization to perform SLAM, both on the current observation and the history of all previous observations. Intuitively, the landmarks and robot locations are represented as nodes in the graph, where adjacent nodes are connected by an edge which contains the odometry reading between the two locations. These edges represent the constraints between the locations, and the trick is to optimize the edge constraints to estimate the maximum likelihood pose and landmark positions. Note that the graph-based model is in essence a full SLAM problem solution, however, recent developments [61] include an incremental version of the graph-based model. Generally, since there are highly efficient optimization techniques available to solve sparse nonlinear optimization, graph-based SLAM [62] is the method of choice for building large-scale maps.

3.3 SLAM implementations

3.3.1 Filter-based SLAM

SLAM algorithms can be categorized into two groups. The classical methods use Bayes-based filter approaches [61], whereas the new methods make use of graphs [62]. Two of the most significant implementations following the Bayes-based filter methods currently available as packages are GMapping [20, 22] and HectorSLAM [33, 32].

While GMapping was developed in 2007, it is still one of the most commonly used system [43]. It uses 2D lidar data to build grid maps using a particle filter approach [20]. However, it was found that the final obtained map did not correctly represent the real map of the environment, as for example the orientations of the hallways were not realistic [16]. HectorSLAM integrates laser scan matching feature with the 3D navigation method based on Extended Kalman Filter (EKF) state estimation [33]. HectorSLAM does not make use of odometry data, which makes it more noisy [43], but alternatively it may have an advantage in environments with pitch and roll characteristics [55]. Furthermore, it does not provide loop closure capabilities to avoid inaccuracies.

3.3.2 Graph-based SLAM

Noteworthy available implementations of the graph-based approach include KartoSLAM [34], RTAB-Map [36], Cartographer [29], and SLAM Toolbox [43]. Graph-based methods represent the map as a graph, where each node represents a pose of the robot. Those are connected by arcs that represent the motion between these poses. A pose-graph contains a set of robot poses connected by nonlinear constraints that are retrieved using the observations of features that are common to nearby poses [34]. However, the optimization of pose-graphs can be a bottleneck for mobile robots, due to computation time scaling quadratically with the size of the graph. To prevent this, KartoSLAM makes use of Sparse Pose Adjustment (SPA) for scan matching and loop-closure procedures [34]. A proposed improvement that, just like KartoSLAM is based on SPA, is the SLAM Toolbox, which provides not only accurate mapping algorithms, but also a variety of other tools [43]. Cartographer also uses lidar data for a graph-representation, where the front-end does scan matching, building trajectory and submaps, and the back-end does the loop closure procedure [29]. However, it is abandoned by Google and no longer maintained. Contrary to aforementioned lidar-based approaches, RTAB-Map, abbreviated from Real-Time Appearance-Based Mapping, is a feature-based approach that builds a point cloud map as graph, with the position and orientation captured image at every pose as nodes [36]. It can be implemented with 3D lasers, stereo vision, or RGB-Depth cameras [55]. RTAB-Map is also the choice for this study because it is well-maintained and contains the interesting features such as Visual Odometry and loop closure.

3.3.3 Visual SLAM

Visual SLAM, where visual sensors are utilized to map and localize an environment, has become a popular and significant subject in recent years [59]. Visual SLAM consists of three main

steps, initialization, tracking and mapping [59]. In the first step, the global coordinate system is defined and an initial estimate is made of the position of the robot in the initialized map. This initialization is then approximated during the tracking step, when the motion through the environment is tracked by determining the changes in the feature positions and estimate the robot's pose based on that. The Feature Extraction step determines what exactly are features. These are distinctive features from the image, such as corners or edges or more complex features retrieved using a feature detection algorithm. Then the features of adjacent image frames are compared in the Feature Matching step. Once matched, an estimation is made of the camera's motion based on the changes of the features. This step is known as Visual Odometry (VO) [48]. Lastly, in the mapping step a 3D map of the environment is built. To potentially improve the map as good as possible, two additional steps are performed: loop closure and optimization. Visual frames offer a lot of information for loop closing, which often relies on visual bag-of-words. Images are reduced to visual 'words' by the feature descriptors, whereafter the similarity of two images can be determined by calculating the product between the word vectors containing the weighted word frequency. If the images are similar enough, that is when the robot has revisited a location, the loop is closed. This results in adjusting the map to correct for the drift. In the optimization step, the reprojection error between the observed and predicted feature positions is minimized to obtain a more accurate map.

3.4 VisualSLAM methods

Apart from RTAB-Map, all aforementioned graph-based approaches like KartoSLAM, Cartographer and the SLAMToolbox require a lidar input, be it in 2D, 3D or the additional odometry data that can be used to help the SLAM approach compute motion estimation. However for visualSLAM, while various open-source approaches exist, not many can easily be used on a robot [38]. One of the first approaches in the visual SLAM domain was introduced in 2007, under the name of MonoSLAM [8], which was equipped with a monocular camera to recover the trajectory. Over the years, the reconstruction methods became more advanced, for example by including global optimization techniques and loop closure detection modules [64]. Also the sensors were changed to provide richer information about the environment, and stereo and RGB-D cameras or visual-inertial odometry were used. For this study, the methods are limited to approaches that do not suffer from scale drift, and are therefore able to estimate the correct scale of the environment to be mapped. Some of the currently most significant implementations comparable with RTAB-Map discussed previously, are ORB-SLAM and Elbrus SLAM. ORB-SLAM2, that can be used by either stereo camera or RGB-D camera, is a graph-based SLAM approach that uses Bundle Adjustment (BA) to optimize the map when a loop closure is detected [47]. As the name suggests, the used features are found using Oriented FAST and Rotated BRIEF (ORB) [53]. The improvement ORB-SLAM3 extends its use by being able to perform visual, visual-inertial and multimap SLAM with monocular, stereo and RGB-D cameras, using pin-hole and fish-eye lens models [7]. Elbrus SLAM [1] from NVIDIA achieves state-of-the-art performance by combining Visual Odometry (VO) and SLAM. VO estimates the camera position relative to its start position, which is used by SLAM to improve the predictions. This way, when the current scene was already previously seen, the obtained poses can be corrected, if necessary. Inertial Measurement Unit (IMU) data can also be used when VO is not able to make an estimation, for example due to bad lightning, or when the camera is obstructed. This addition can lead to significant performance improvements in cases of poor visual conditions. In the experiments, the robot was able to find multiple products in various environments with unpredictable dynamic situations.

[1] `https://docs.nvidia.com/isaac/packages/visual_slam/doc/elbrus_visual_slam.html`

CHAPTER 4

Related work

In this chapter, the related work of multiple challenges of this research will be discussed. The primary focus will be on various feature detection techniques, that can be integrated with a visual SLAM approach. First of all, the importance and the first approaches of feature detection, specifically when using visual SLAM are introduced, whereafter the currently available feature detectors are discussed. These feature detectors are categorized in binary features, floating-point features and deep-learning based features.

4.1 Feature Detection

Visual SLAM is mostly based on determining the geometric relationships between the obtained sequences of images and the surrounding environment by feature extraction and matching [10]. This makes the detection of features a crucial step [57]. Whereas visual features were previously extracted using corner feature detectors, such as Harris [26] and Förstner [17], the development of invariant feature description methods, such as SIFT [41], the applicability of visual SLAM has been improved significantly [10]. This is because both the extraction and matching of the keypoints in the image are able to be performed with limited image deformation and illumination changes. However, since these performance improvements require significant computations and are therefore accompanied by a limitation in the efficiency, more feature detectors have been developed to regard the efficiency [10]. These include feature detectors such as SURF [3], BRISK [40] and ORB [53]. In this chapter, the currently available feature detector that are able to be integrated with a visualSLAM approach will be discussed. Furthermore, three categories are distinguished, namely binary features, float features and learned features, since generally all feature types can be placed in one of these categories [39].

4.2 Binary features

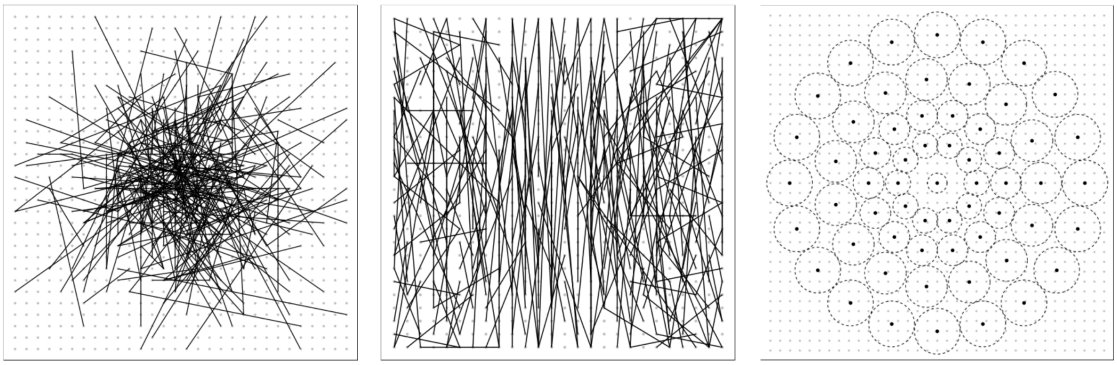
Binary features are characterized by the following common principles [28]. First of all, the descriptor corresponding to the binary keypoint is created by comparing the intensities in pairs. Secondly, each bit in the descriptor represents the outcome of one single comparison. Thirdly, the sampling pattern remains constant, except for potential changes in scale and rotation. And lastly, the similarity between descriptors is measures using the Hamming distance [25]. On top of these common principles, each binary feature is characterized by different properties compatible with its design goals.

BRIEF (Binary Robust Independent Elementary Features) is the simplest of methods in this section. A smoothing kernel is applied with noise sensitivity, whereafter every keypoint is described by a binary feature detector [6]. The sampling pattern can consist of either 128, 256 or 512 comparisons, which results in 128, 256 or 512 bits respectively. The sampling points are randomly

selected from an isotropic Gaussian distribution which is centered at the feature location (see Figure 4.1a for an illustration).

ORB (Oriented FAST and Rotated BRIEF) builds on the well-known Features from Accelerated Segment Test (FAST) keypoint detector [52] and the previously introduced BRIEF descriptor [6]. FAST is a high-speed keypoint detector that uses machine learning to classify whether a certain point is a interesting corner point. It is faster than other existing corner detectors, and is able to reach high levels of repeatability under large aspect changes, but it is not robust to high levels of noise and dependent on a threshold [52]. ORB is an improvement of BRIEF in the sense that it overcomes the lack of rotation invariance. This is achieved by computing a local orientation by utilizing an intensity centroid [53]. This is determined by averaging the intensities of the pixels that are within a specific area around the feature. Then the orientation is determinated by the vector between the mentioned centroid and the feature itself. The sampling pattern is performed using 256 pairwise intensity comparisons, however, ORB, unlike BRIEF, constructs the pattern using machine learning techniques. The aim is to maximize the variance of the descriptor while simultaneously minimizing the correlation under different orientation changes (see Figure 4.1b for an illustration).

BRISK (Binary Robust Invariant Scalable Keypoints) adds another functionality on top of ORB, namely being both scale and rotation invariant [40]. BRISK utilizes the AGAST corner detector [44], which is a faster version of FAST for the feature detection, which is done in a scale-space pyramid to realize scale invariance. Furthermore, non-maxima suppression and interpolation across multiple scales is performed to prevent duplicates. Contrary to BRIEF and ORB, a symmetric pattern is used to describe the features. The sampling takes place in concentric circles around the feature, where each sample points represents a Gaussian blur of the surrounding pixels. The degree of blurring (the standard deviation) increases as the distance from the center of the feature grows (see Figure 4.1c for an illustration). The orientation is determined by using several long-distance sample point comparisons, that capture vector displacements weighted by intensity differences. Averaging these reveals the dominant gradient direction. The sampling pattern is scaled, rotated and built into the descriptor using 512 short-distance sample point comparisons, representing local gradients and shape [40].



(a) (b) (c)

Figure 4.1: Example patterns of the (a) BRIEF, (b) ORB, and (c) BRISK feature type algorithms. Courtesy from [28].

4.2.1 Comparison

These feature detection algorithms have been compared in three categories, namely non-geometric transforms, affine image transforms and perspective transforms [28]. Non-geometric transforms consist of transformations that are image-capture dependent and do not rely on the viewpoint. In this category, even though BRIEF has a noticeably lower precision than ORB and BRISK,

it outperforms them, which makes BRIEF favorable. This can be explained by its fixed pattern. Also, the authors note that the lower precision is caused by having either a less restrictive matching criteria or feature distinction, which also does not make it surprising that the precision of BRIEF suffers slightly. Affine image transforms consist of image plane rotation and scaling. In this category, ORB and BRISK performed well, whereas BRIEF performed rather poorly. This is expected because BRIEF is neither rotation and scale invariant. Furthermore, since ORB only has the rotation invariant attribute, whereas BRISK has both, BRISK performs the best of all binary feature detectors. Perspective transforms are achieved by changes in viewpoint. Surprisingly, even though BRIEF has a limited complexity, it has a slight lead in recall and matching, but the authors note that this may be explained by the upright nature of the dataset. Additionally, the precision of BRISK is better than the other two, which points to the descriptiveness of BRISK. Therefore, even though BRIEF has the lowest compute and storage requirements, it does not always perform the worst. Whereas BRISK requires significantly more computational resources, and frankly performed best in these experiments, but may not always outperform other binary features. This highlights the importance of leveraging knowledge about the data and specific use case [28].

4.3 Floating-point features

Floating-point features are, contrary to binary features, not represented by a single bit for every comparison, but by a floating-point value, which gives them potentially more descriptive power.

SIFT (Scale Invariant Feature Transform) is one of the most commonly used feature detection algorithm in vision tasks [27], and goes through four main stages to obtain the keypoints [41]. The first step is to search for a set of stable features over multiple sclaes using a Difference-of-Gaussian (DoG) function [5] to detect the scale-space extrema. Since the local extrema are most interesting, a sample point is only selected if it is larger or smaller than all the neighboring points. Note that there are eight neighboring points in its current scale and nine neighboring points in the two adjacent scales. Next, a detailed model is fit to determine the location, scale and ratio to localize keypoints accurately. In this step keypoints are filtered to guarantee stability. Thirdly, the orientation is assigned, which consists of assigning one or more orientations to each keypoint based on the local image gradient directions. This enables determining a descriptor for each detected point based on the local image information. The description stage consists of sampling image gradient magnitudes and orientations and creating orientation histograms, which is overlayed by a Gaussian weighting function to account for gradients less affected by positional changes. The obtained histograms can then be transformed and normalized to a 128-dimensional feature vector for every keypoint. The creation of the keypoint descriptor is visualized in Figure 4.2. SIFT is carefully designed to avoid problems in boundary changes, which is evident in its performance, however, its high dimensionality affects the computational time significantly [27].

SURF (Speeded-Up Robust Features) was developed as alternative to the well-established SIFT. While the aim was to come close to its competitor in terms of repeatability, distinctiveness and robustness, the improvement lay in the ability to be faster to compute and match. For the detection part, interest regions in the form of blob-like structures are detected based on the determinant of the Hessian matrix, which is a square matrix that consists of second-order Gaussian derivatives. A blob response map over different scales is obtained by approximating the second order Gaussian derivatives, specifically when the derivatives responses are high in two orthogonal directions or if the determinant has a local maxima. Performing a non-maximum suppression assures that unreliable points are filtered out. Once detected, description is done by creating a square window around the detected keypoint, which is positioned at its center and aligned with its main orientation. This interest region is then divided into smaller sub-regions, for which wavelet responses are computed, which in turn are multiplied by a Gaussian filter to improve the found keypoints in robustness. These can then be summed and combined into a

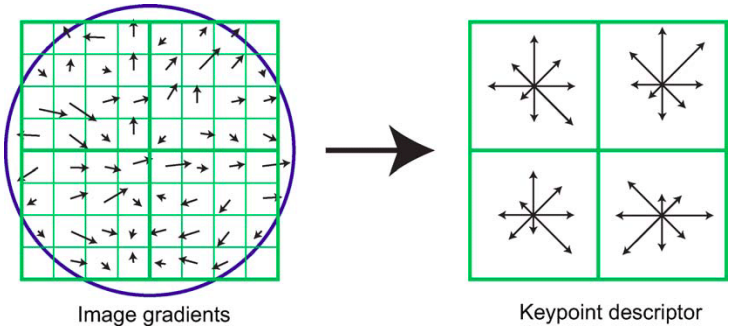


Figure 4.2: A simplified schematic representation for computing the SIFT descriptor by firstly computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location weighted by a Gaussian (left), which are then accumulated into orientation histograms (right). Courtesy from [41].

normalized feature vector with 64 dimensions [3]. The creation of the descriptor is visualized in Figure 4.3.

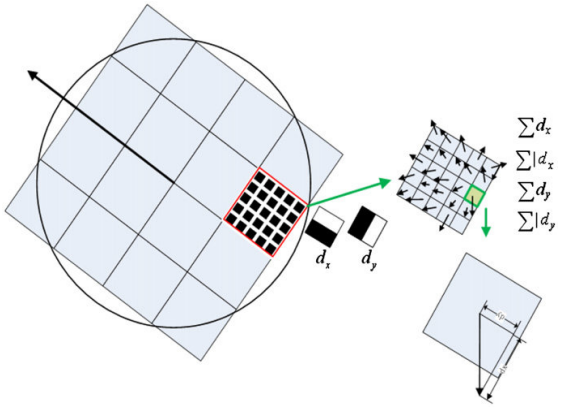


Figure 4.3: A simplified schematic representation for computing the SURF descriptor by using an oriented grid. Courtesy from [27].

KAZE (Japanese for *wind*) is a multi-scale feture detection and description approach which operates in the nonlinear scale space. Exactly like the previously discussed SIFT and SURF approaches, multi-scale features are found, however, SIFT and SURF find these by filtering the input image with Gaussian blurring, whereas KAZE makes use of nonlinear diffusion filtering [67]. An advantage of nonlinear diffusion filtering over Gaussian blurring is that Gaussian blurring no respect has for natural boundaries of objects, meaning that details and noise are smoothened in the same manner. Nonlinear diffusion filtering on the other hand is able to blur noise out but keep image details and object boundaries unaffected, which leads to obtaining better localization accuracy and distinctiveness. These effects are visualized in Figure 4.4.

The four main steps discussed with SIFT are also appropriate here. Firstly the nonlinear scale space is computed using a nonlinear diffusion filtering technique combined with a conductivity function. The filtered images are processed using an Additive Operator Splitting (AOS) method [14]. To detect the keypoints, the response of the scale-normalized determinant of the Hessian at multiple scale levels is computed, and to localize the keypoints the maxima in scale and spatial location are found by making use of non-maximum suppression. The orientations are assigned according to an estimation of a centered circular area around the keypoints. Lastly, the descriptors are computed using a modified non-linear scale-space version of the SURF descriptor, leading to a 64 dimensional feature vector. Identical to the other discussed feature detectors, this feature vector is then normalized to an unit vector for robustness [1] [27].

Figure 4.4: Comparison between Gaussian or linear (top row) and nonlinear (bottom row) diffusion scale space for various evolution times t_i. Courtesy from [1].

4.3.1 Comparison

Extensive comparisons between these floating-point feature detectors and descriptors are made [1]. Whereas in the binary features category there were differences in invariancy, in this category the dicussed feature detectors are all rotation and scale invariant. However, still some differences could be observed in image rotations and scale changes experiments [27]. In the experiments, the metrics used are precision, recall and the number of inliers. Only image rotations were performed best by KAZE followed by SIFT, for only scale changes SIFT beat all the other detectors in all metrics, and combined rotation and scale changes resulted in the best performance by SIFT but both KAZE and SURF obtained a good performance as well. As for the perspective changes, there were multiple datasets evaluating changes of viewpoint, but on one dataset SIFT performed best, whereas for another dataset KAZE took the lead. Unsurprisingly, KAZE performed best when investigated under the blur effects, as KAZE takes natural object boundaries into account. All in all, several observations could be made. While the claim from the KAZE authors that it could surpass SIFT is supported by the results, SIFT still outperforms in pure scaling. The performance under JPEG compression effect is favorable for KAZE, which is the best performer in all metrices.

4.4 Deep learning-based features

The last category of features are the deep learning-based feature, which could be in either a supervised, self-supervised or unsupervised manner. Either way, the task is often converted into a regression problem, which is trained in a differentiable way under various transformations and imaging condition invariance constraints [42]. Supervised approaches rely on human annotations to obtain the model, however, these are hard to properly define and are therefore more prone to being restrictive. This could lead to preventing the network to find and propose new keypoints. Contrarily, self-supervised and unsupervised approaches do not require human-labeled annotations, and only the geometric constraints between two images suffice. Generally, the training for the feature description and matching happens simultaneously and the feature detection is integrated into the matching pipeline directly. This could improve the matching performance as the entire procedure can be optimized end-to-end. In this study, solely self-supervised and unsupervised approaches will be discussed.

SuperPoint SuperPoint is a self-supervised fully convolutional model that inputs the full-sized images and jointly computes the locations of the pixel-level interest points and its corresponding detectors in a single forward pass [9]. There is a single shared encoder that enables the input image dimensionality to be processed and importantly reduced. Then, the architecture is divided up into two decoders, namely one for interest point detection and one for interest point descriptors. While the two decoders each learn their own task specific weights, most of the parameters of the network are shared between both tasks. Note that this is different from the traditional

systems, as then the interest points are firstly detected, whereafter the descriptors are computed, meaning that they lack the ability to share computation and representation across both tasks. The SuperPoint architecture is visualized in Figure 4.5. Note the decoder architectures for both tasks.

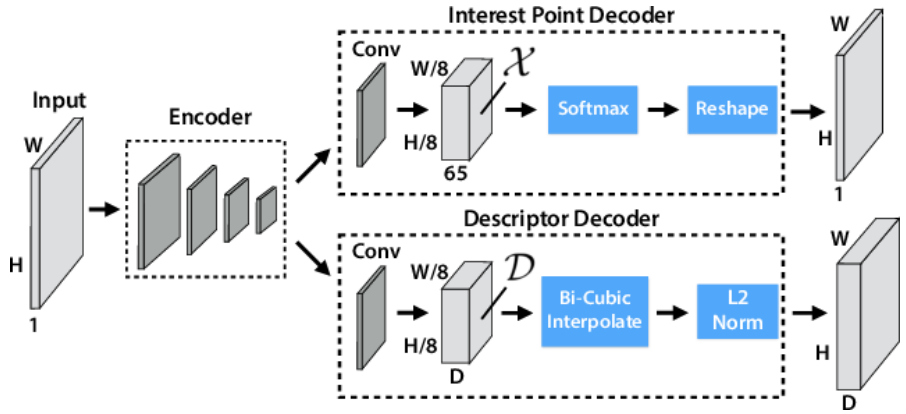


Figure 4.5: Superpoint architecture, where both decoders operate on a shared and spatially reduced representation of the input image. Courtesy from [9].

Furthermore, the authors propose in their self-supervised training approach a pre-training step and a self-labeling step using a novel procedure that they call Homographic Adaptation. Firstly a large dataset of pseudo-ground truth interest point locations in real images is constructed, which is supervised by the interest point detector itself, meaning that a large-scale human annotation effort is not necessary. These pseudo-ground truth interest points are generated by training a fully-convolutions neural network on a synthetic dataset, resulting in a trained detector the authors name MagicPoint. While MagicPoint performed surprisingly well on real images, especially when considering domain adaptation issues, it misses many potential interest point locations. To repair this, the multi-scale, multi-transform technique Homographic Adaptation was developed. This module warps the input images and so achieves self-supervised training while boosting detection repeatability.

R2D2 R2D2 is a fully-convolutional model that is designed to enable end-to-end optimization for both feature extraction and description. An explicit distinction is made between reliability and repeatability, of which a detected feature must both comply to [51]. The authors claim that both characterizations are complementary aspects that must be predicted separately. In Figure 4.6 toy examples illustrate the crucial difference between repeatability (visualized in the second column) and reliability (visualized in the thrid column) of two given input images. Whereas only the pixels near the black region are regarded as repeatable in the left image, for the reliability the pixels surrounding the black region are all equally confident. On the other hand, the right input image shows an example of a case whereby there are a lot of salient and thus repeatable pixels, namely all the squares in the checkerboard, but due to self-similarity none of the pixels is discriminative, resulting in no pixels being regarded as reliable. Therefore, the proposed network R2D2 is a new learning-based feature extraction method that, in contrast to existing deep-learning based methods, learns both keypoint repeatability and a confidence for keypoint reliability [51]. As a backbone the network L2-Net is used [63], and the network is furthermore trained via self-supervision using a combination of synthetic and real images. Also, style transfer methods are applied to increase robustness against drastic illumination changes.

D2-Net In D2-Net a single CNN is used that plays a dual role; it is both a dense feature descriptor and a feature detector simultaneously. Note that while the proposed is also a detect-and-describe approach, the major difference is that in D2-Net the detection stage is postponed. Instead of utilizing low-level information to detect the features early in the pipeline, firstly a set of feature maps is computed via a CNN, which is then used to both compute the descriptors

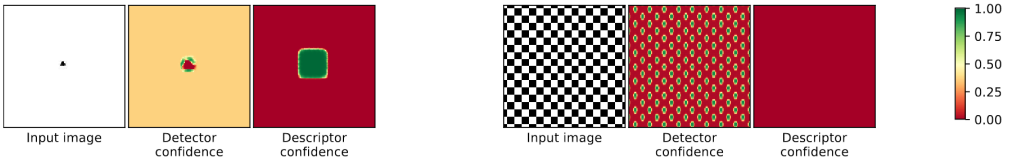


Figure 4.6: Toy examples that illustrate the distinction and specific difference between repeatability (second column) and reliability (third column). Courtesy from [51].

and detect the keypoints. Thus, the locations of the detections correspond to the pixels of the descriptors that are distinctive and therefore appropriate for matching. The D2-Net architecture is visualized in Figure 4.7. The network finetunes a pretrained VGG network [35]. While the authors acknowledge the fact that their approach is less efficient due to its dense nature of the extraction, and that using higher-level information may lead to less accurate keypoints, they show that D2-Net achieves state-of-the-art performance and even the less accurate keypoints are still more robust and nevertheless accurate enough for visual localization [12].

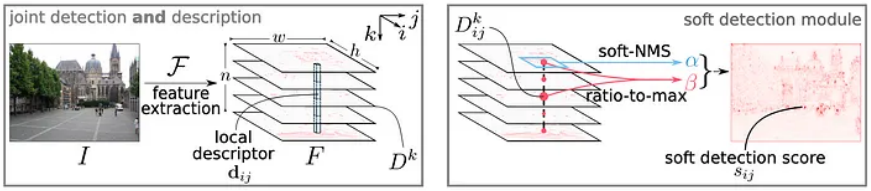


Figure 4.7: A visualization of the D2-Net, which uses a feature extraction CNN $\mathcal{F}$ to extract feature maps that a) obtain local descriptors by going through the n feature maps D^k at location (i, j), and b) obtain detections by performing a non-local-maximum suppression on a feature map followed by a non-maximum suppression across each descriptor. Courtesy from [12].

4.4.1 Comparison

In this section, three deep-learning based feature detectors have been discussed, which all utilize the one-stage approach to perform the detection and description tasks simultaneously. Both SuperPoint and D2-Net encode the images in a feature map of 1/8 of the size of the original image, whereas R2D2 preserves the size of the original input image. This leads to a better strict pixel accuracy, but the obtained keypoints are mostly from large receptive fields. Furthermore, SuperPoint and R2D2 both share a representation from the deep neural network to learn a dense feature descriptor and a feature detector, but they are trained independently, whereas in D2-Net all parameters between the detection and description tasks are shared and are optimized simultaneously. R2D2 regards the repeatability and reliability as two separate entities that need to be optimized, whereas SuperPoint and D2-Net do not make such distinction. This distinction results in the experiments of [57] of R2D2 outperforming the other two feature detections significantly in repeatability in all transformations, namely viewpoint perspective, zoom and rotation, luminosity, blur and compression, with D2-Net performing considerably better than SuperPoint except for the luminosity and compression transformations. As for the 3D reconstruction experiments, SuperPoint and R2D2 are able to perform better in the number of verified image observations per sparse points, but the completeness of the reconstruction the opposite is the case. This may demonstrate that SuperPoint and R2D2 find mostly discriminative keypoints and therefore may ignore keypoints are that not distinct enough. The accuracy of the reconstruction, the reprojection error, is the worst for D2-Net.

4.5 Discussion

In this chapter, three categories of feature detection approaches and various corresponding existing feature detection methods are highlighted. A distinction is made between binary features, floating-point features and deep-learning based features. The binary feature detectors are the simplest of the three categories and are generally outperformed, but due to the performance trade-off it could be a viable choice when the application provides robustness against outliers. Or since they are the most-efficient, they could be a better choice than the more time and memory demanding approaches. Whereas the binary and floating-point features make use of a two-stage approach, also called detect-then-describe, where firstly the keypoints are detected whereafter the descriptors can be described, the deep-learning based features are found by a detect-and-describe approach, which only consists of one stage. In the one-stage approach, there should be more consistency between the keypoint detections and the keypoint descriptors. However, there is no universal and optimal approach, since the best choice is generally dependent of the specific task, the setup and the existing distortions in the available data. In this thesis, we will compare the different feature detectors on their usability in the visual SLAM algorithm RTAB-Map.

CHAPTER 5

Method

In this chapter the approach of performing Visual SLAM is discussed. Firstly, the utilized hardware and software is introduced, whereafter the implementation details are explained. This includes the implementation of the addition of the software package RTAB-Map and the implementation of an evaluation method based on the fiducial principle.

5.1 Robot Spot

Spot is a quadrupled robot developed by one of the world's leading robotics companies, Boston Dynamics. It is equipped with advanced sensors and cameras, and able to collect and analyze data.

Figure 5.1: The robot Spot from Boston Dynamics, with its base platform.

The base platform [1], as visualized in Figure 5.1, provides advanced mobility and perception and simultaneously collects 2D and 3D information with on board sensors. Spot's mobility is enhanced by the fact that it has legs, meaning that Spot is able to traverse and navigate through challenging environments, such as stairs, as opposed to for example wheeled robots. Spots movement is achieved by the use of 12 hydroscopic motors in this legs. However these 12 degrees of freedom can make the robot more susceptible to drifting, because more variability to the movement is introduced, so it is crucial properly calibrate to mitigate the effects of drifting. Furthermore, Spot has 4 sensing modules, each consisting of a pair of stereo cameras. However, these cameras are faced slightly downwards, because these cameras are intended for obstacle avoidance; this configuration is less favorable for mapping, which may result to data being lost or cut off.

[1] `https://www.bostondynamics.com/products/spot`

In addition to the base platform, provided payloads can be added to enhance Spot's sensing and data processing capabilities, for example on the levels of perception (SpotCAM), computation (SpotCORE), autonomy (SpotEAP), integration (SpotGXP) and manipulation (SpotArm). The utilized Spot of the Robot Learning Group has added the SpotArm and a SpotCORE. This arm can be used to open and close doors or pick up objects. This makes the robot even more versatile. This Spot is shown in Figure 5.2.

Figure 5.2: The available Spot from the Robot Learning Group at the University of Osaka.

5.2 Hardware

In addition to the by Boston Dynamics provided hardware, an additional payload has been integrated into Spot. In particular to solve the issue of the downwards facing cameras that are not as helpful for mapping as for obstacle avoidance. The total payload consists of the SpotCORE, LiDAR sensor and RGB-D camera. The SpotCORE is an additional CPU processor, which enables software to run locally on the robot, resulting in high-bandwidth low-latency connections. The specifications are an i5 Intel 8th generation motherboard, 16 GB of RAM memory, and 512 GB of primary storage. The newer variant of the SpotCORE, SpotCORE I/O, does have access to both a CPU and a GPU, which enables more training opportunities on the SpotCORE itself. However, for this study there was no access, meaning that training had to be done on an external GPU, that could be connected to the SpotCORE. The SpotCORE comes preconfigured with Ubuntu 18.04. The LiDAR sensor is Velodyne VLP-16, which uses 16 laser beams to provide 360 degrees coverage and can capture up to 300.000 points per second with a range of up to 100 meter. Note how the Velodyne is placed on the robot. While this high placement enables the sensor to capture a proper 360 degrees view, it could lead to a more wobbly ride, which may eventually result to drift. The RGB-D camera is the ASUS Xtion Pro Live sensor, which is based on the PrimeSense technology. This makes use of a cheaper alternative to stereo camera systems, namely structured light technology in the near infrared spectrum for the depth measurements, and also includes an RGB sensor. Furthermore, there are hardly CPU requirements, as the data processing is done by PrimeSense's System-on-a-Chip (SoC). The distance of use is between 0.8 and 3.5 meter.

5.3 Software

5.3.1 ROS

The most important software tool used in this implementation is the Robot Operating System (ROS). ROS is, unlike the name suggests, not a operating system but a software framework that provides a framework for communicating with multiple processes [50]. ROS is used extensively in both research and industry, and has a large community of developers that contribute to its development. ROS consists of multiple fundamental concepts, namely nodes, topics and messages. A node is a process that performs the computations. Therefore, this is also called the software module. Nodes receive the data to be processed in the form of messages from topics, which are named buses over which nodes exchange messages. And a message is a strictly typed data structure. A node is able to subscribe and publish to multiple topics simultaneously, and there may be multiple concurrent publishers and subscribers to a single topic.

5.3.2 RTABMap

In this thesis, the primarily used software is the package RTABMap-ROS, which is a ROS wrapper for RTAB-Map, Real-Time Appearance-Based Mapping [36] [38] discussed in section 3.3. RTAB-Map is a RGB-D SLAM approach based on a global loop closure detector with real-time constraints, and it was chosen due to its versatility. While it was originally proposed as an solely appearance-based solution, it has been well-maintained and evolved as a full online graph-based visual SLAM approach with notable practical features such as online processing, robust and low-drift odometry, robust localization, practical map generation and exploitation and multisession mapping. RTABMap can be used both using the default visual odometry and external odometry, resulting in the visual information only being used for loop closure detection [60] and map reconstruction. Loop closure is done using a visual bag of words approach [37], where the vocabulary is built during the process. As the RTABMap package is constantly being developed, it has been updated to incorporate new and state-of-the-art feature detectors, which enables the possibility to make comparisons and gain insights.

Figure 5.3 shows the overview of the ROS node rgdb_odometry functionality where this study heavily relies on. Note that input TF is the position of the camera in relation to the base of the robot and the output TF is the odometry transform of the base of the robot. This pipeline is feasible for both a RGB-D camera and a stereo camera, but in case of the stereo camera the calculation of stereo correspondences is required. However, in this thesis the RGB-D is assumed.

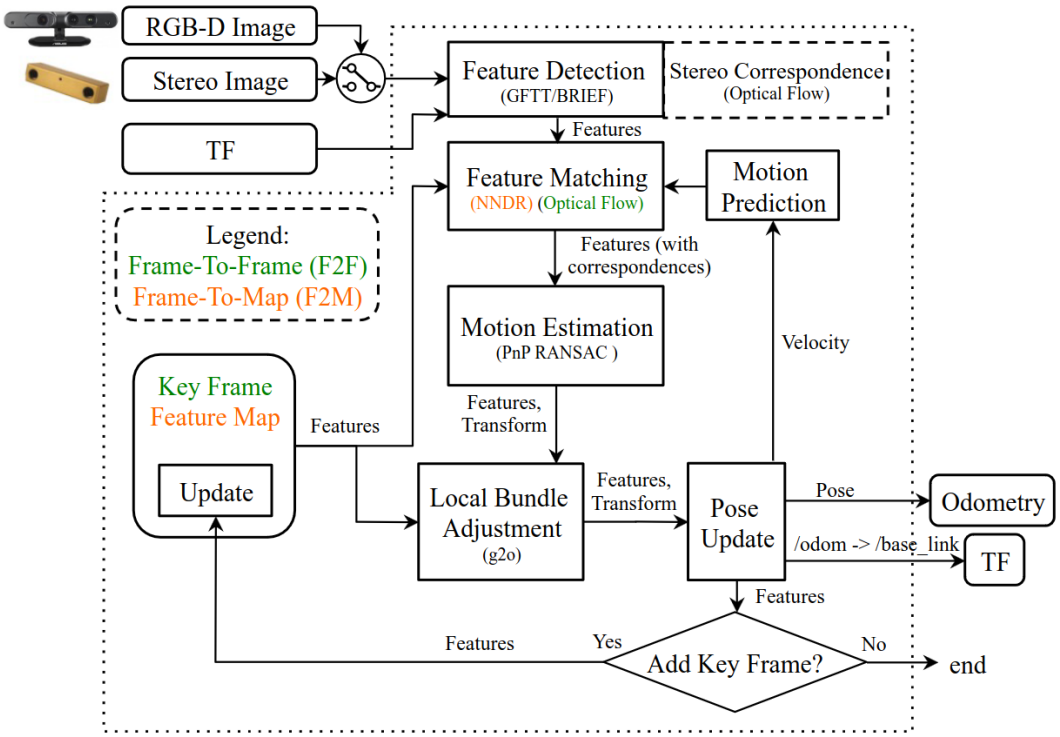


Figure 5.3: Block diagram of rgbd_odometry and stereo_odometry ROS nodes. Courtesy from [36].

In this thesis, the main focus will be on the feature detection step of the visual odometry of RTABMap. Visual Odometry (VO) is a technique that aims to estimate the position ans orientation of a robot by using the visual input, so the motion of the camera relative to the environment [59]. In case of the RTABMap, two visual odometry flavours are implemented: Frame-To-Map (FTM) and Frame-To-Frame (FTF). Whereas FTF compares the new frame with the last key frame, FTM compares the new frame with the local map of features that are created from the past key frames. The visual odometry process works as follows [38]: For every incoming frame, feature detection is performed using GoodFeaturesToTrack algorithm. However, all feature types available in OpenCV are supported by RTABMap. When receiving RGB-D images, the depth map is used as a mask to avoid feature detection with invalid depth values. The detected features are then matched by either matching the current frame features using by default BRIEF descriptors against a FeatureMap using nearest neighbor search (F2M), or by matching the current frame features with optical flow against the key frame, without the need for descriptors (F2F). This difference is also shown in Figure 5.3 below Feature Detection. Then, a motion model predicts where either the Key Frame features (F2F), or the Feature Map features (F2M) should in the current frame, whereafter the transformation of the current frame according to the features in either the Key Frame (F2F), or the Feature Map (F2M) is calculated in the motion estimation step. This is followed by the local bundle adjustment step, in which the resulting transform in refined on either the last Key Frame features (F2F), or on the features on all the key frames in the Feature Map (F2M). Lastly, both the poses as the Key Frame (F2F) and Feature Map (F2M) are updated.

As mentioned in the previous section, in the feature detection step, RTABMap allows for all OpenCV feature types to be used. As of now, the following CV feature detectors have been integrated into RTABMap. Some of the implemented feature detection algorithms are also discussed in chapter 4, but they are here all shortly summarized again as a reminder. Note that when there are two feature types listed together, for example FAST/FREAK, it means that that algorithm is configured to use a combination of both feature types, so in this case FAST is used as a feature detector and FREAK is used as a feature descriptor. Since every feature type has its strengths and weaknesses, combining two types might give an increase in performance.

SURF (Speeded Up Robust Features) is a feature transform algorithm based on the concept of scale-space representation, so the image is analyzed at different scales to detect and describe the features. Therefore, it is scale-invariant, and also robust to changes in rotation and illumination.

SIFT (Scale-Invariant Feature Transform) is, as the name suggests, also a scale-invariant feature transform algorithm that detects and describes local features. The principle of SIFT is detecting and describing the key interest points in an image, and matching those of that of another image.

ORB (Oriented FAST and Rotated BRIEF) is an algorithm that uses oriented FAST keypoints and rotation-invariant BRIEF descriptors to detect and describe features. Since ORB is designed to be fast and efficient, it is well-suited for practical applications.

FAST/FREAK combines the FAST feature detector with the FREAK descriptor. FAST (Features from Accelerated Segment Test) is a corner detection algorithm, which is very fast but not robust to high levels of noise. FREAK (Fast Retina Keypoint) is a binary descriptor intended for fast matching.

FAST/BRIEF combines the FAST feature detector with the BRIEF descriptor. BRIEF (Binary Robust Independent Elementary Features) is, like FREAK also intended for fast matching, but differs mainly in the sampling pattern and descriptor size.

GFTT/FREAK combines the GFTT feature detector with the FREAK descriptor. GFTT (Good Features To Track) is a combination of the Harris and GFTT corner detection algorithms, that calculates the local autocorrelation of the intensity of the image.

GFTT/BRIEF combines the GFTT feature detector with the BRIEF descriptor.

BRISK (Binary Robust Invariant Scalable Keypoints) uses AGAST, a faster version of FAST, for the feature detection and has similar properties to SIFT, since it is also scale and rotation invariant. However, the sampling pattern differs, because BRISK uses concentric circles.

GFTT/ORB combinest the GFTT feature detector with the ORB descriptor.

KAZE (named after the Japanese word for *wind*) is a feature detection and description method that is able to operate in a nonlinear scale space (as opposed to the Gaussian scale space in for example SIFT or SURF), which makes it more robust to challenging image conditions.

ORB-OCTREE is a modification of the ORB feature detector and descriptor that uses an octree structure, which enables faster matching.

SuperPoint is a deep learning-based feature detector that uses self-supervised learning. Synthetic data is used to pretain the detector, and then test time augmentation is used to transfer to real data.

SURF/FREAK combines the SURF feature detector with the FREAK descriptor. Since both algorithms are designed with efficiency in mind, it is effective in large-scale images.

GFTT/DAISY combines the GFTT feature detector with the DAISY descriptor. DAISY (no acronym given) is very efficient to compute densely, and excells in begin robust to deformations.

SURF/DAISY combines the SURF feature detector with the DAISY descriptor.

PyDetector enables using a feature detection method with an existing implementation. This path of this implementation can be added and then this implementation will be used.

The method used by default is GFTT/ORB if the xfeature2d module of OpenCV is not available, and otherwise GFTT/BRIEF. Furthermore, there are restrictions for using certain feature detectors. For example, in order to utilize SuperPoint, OpenCV must be built with both CUDA and PyTorch support.

5.4 Implementation

The implementation of this thesis consists of two main parts. While the focus was to draw insights in the currently available feature detectors in RTABMap and make an extension by implementing a novel feature detection approach, when making a comparison and an evaluation of the feature detection approaches, there was no ground truth available, which make it clear that an assessment approach of the quality of the obtained map was also a crucial research component. Therefore, in addition to a feature extension, an evaluation method based on the fiducial principle is proposed.

5.4.1 RTABMap addition: R2D2

In this study, the functionality of RTABMap is extended by implementing an additional feature type, R2D2. R2D2 is also globally discussed in section 4.4, but will be discussed to a greater extent in this section. R2D2 stands for Repeatable and Reliable Detector and Descriptor [51]. In the paper, the authors distinguish two methods for interest point detection and local feature description, namely the classical methods and the neural network approach. The classical methods are characterised by a detect-then-describe paradigm, where the interest points are firstly identified and afterwards described using local feature descriptors. While these methods have been widely used in computer vision applications, they are often manually tweaked meaning that they can not generalize well. So recently, the deep-learning methods have emerged and caught up with these techniques by learning repeatable saliency maps for interest point detection and learning descriptors at the detected keypoint locations. Groundbreaking approaches in this category include SuperPoint and SuperGlue. The authors acknowledge the performance and efficiency of the neural network approach, but argue that salient regions are not necessarily discriminative and hence could even harm the performance of the description. Additionally, they claim that descriptors should only be learned if matching with high confidence is achievable. Thus, R2D2 is proposed, a jointly learned keypoint detection and description method together with a predictor of the local descriptor discriminativeness. Crucial is the explicit distinction between **repeatability** and **reliability**, which is claimed to be two complementary aspects that should be predicted separately.

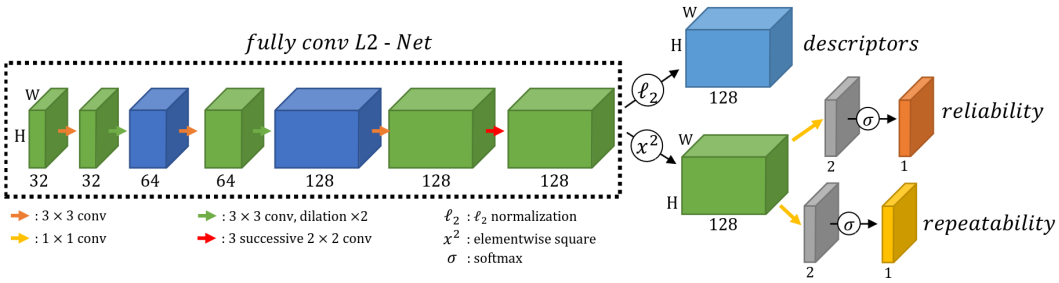


Figure 5.4: Overview of the R2D2 network for jointly learning repeatable and reliable matches. Courtesy from [51].

The fully-convolutional network (FCN) shown in Figure 5.4 has three outputs for an image I of size $H \times W$.

1. a 3D tensor $\boldsymbol{X} \in \mathbb{R}^{H \times W \times D}$; This tensor contains a set of D-dimensional dense local discriptors. These descriptors correspond to individual pixels in the input image I.

2. a *repeatability* heatmap $\boldsymbol{S} \in [0,1]^{H \times W}$; This is a confidence map that is an estimate of the probabilities that a keypoint is repeatable. Since this confidence map is trained to contain only strong and repeatable local maxima, it should only have confidence in sparse and repeatable keypoint locations. The confidence map contains a value for every descriptor $\boldsymbol{X}_{ij}$ at each pixel (i, j), with $i = 1...W$ and $j = 1...H$.

3. a *reliability* heatmap $\boldsymbol{R} \in [0,1]^{H \times W}$; This is a confidence map that is an estimate of the probabilities that a keypoint is reliable. A value for the discriminativeness factor means that a keypoint can accurately be matched with high confidence. The confidence map contains a value for every descriptor $\boldsymbol{X}_{ij}$ at each pixel (i, j), with $i = 1...W$ and $j = 1...H$.

Then, the keypoints are extracted corersponding to the locations that maximize both confidence maps $\boldsymbol{S}$ and $\boldsymbol{R}$. The obtained keypoints and its corresponding descriptors can then be fed into the existing RTABMap framework, as visualized in Figure 5.3.

5.4.2 Evaluation

To assess the obtained map quality of RTAB-Map with various feature detectors and descriptors, normally a ground truth of the area has to be available, but that was not available during this research. Therefore, to determine the performance a fiducial approach is used. Originally used in the context of the RoboCup Rescue league, the fiducials (artificial objects placed in the environment) can function as a simplification of the ground truth of the environment [56], meaning that the ground truth of fiducial locations can be compared with that of the observed fiducial locations. Note that this approach abstract the environment completely, by not necessarily observing information like walls or explored areas, but by relying solely on the observed fiducials. This also makes the performance heavily dependent on the placement of the fiducials, and the exploration in that environment. However, for situations where it is seeked to make a comparison between various methods in the same environment, and thus the same placement, this should not cause issues. The fiducials approach works for both visual SLAM and lidar SLAM. For lidar SLAM, the fiducials are typically cylinders that are cut in half and placed on either side of a wall. These circular fiducials will stand out in the obtained 2D map, which makes it easier to evaluate. For visual SLAM, the fiducials are Augmented Reality (AR) markers.

AR tags and more generally fiducial markers, are highly distinguishable patterns that contain strong visual characteristics [30]. Most of the markers are visually QR-like black and white square images within a black border, as the four points that make up the square allows for camera calibration and marker position calculation [54]. There are various AR tag generation algorithms that each produce different tags and a different way of detecting them. AR tags of three of the state-of-the-art and most commonly used algorithms are visualized in Figure 5.5. The detection strategy consists of two main steps for all three approaches. First of all, the image or environment is searched for a unique feature, such as a quadrilateral shape. Then the identification step validates whether the detected marker is in fact an AR tag. Since all approaches make use of a different design, detection and recognition algorithm, they all different strengths and weaknesses in various situations [54]. For this study the ARTag marker was selected due to its lowest computational cost [30] and the knowledge that the usecase for this study did not require the AR tags to be very robust. ARTag [15] uses a simple detection and recognition algorithm, where image binarization is used to detect and map potential markers with a set of marker patterns.

Figure 5.5: ARTag (left), AprilTag (middle) and CALTag (right) markers examples. Courtesy from [54].

For this research, the ROS package `ar_track_alvar` was used, a ROS wrapper for Alvar, an open source AR tag tracking library, which is known to provide a large amount of flexibility in the style and content of the markers [30] . This package is able to generate AR tags, and identify and track the pose of AR tags. The first step is to generate AR tags that can be placed into the environment. Examples of generated AR tags are shown in Figure 5.6.

Figure 5.6: Various generated AR tags using the `ar_track_alvar` package.

Now, when the robot walks through the environment the `ar_track_alvar` can be run to identify and publish the locations of the identified AR tags. The `ar_track_alvar` package subscribes to the topics `/camera/rgb/image_raw` and `/camera/rgb/camera_info` and publishes the topics `visualization_marker` and `ar_pose_marker` and a transform from the camera frame for every detected marker named `ar_marker_x`, with `x` being the marker id. Those relations are visualized in a graph shown in Figure 5.7. Note that the rectangles represent topics, and the ellipse represents a node.

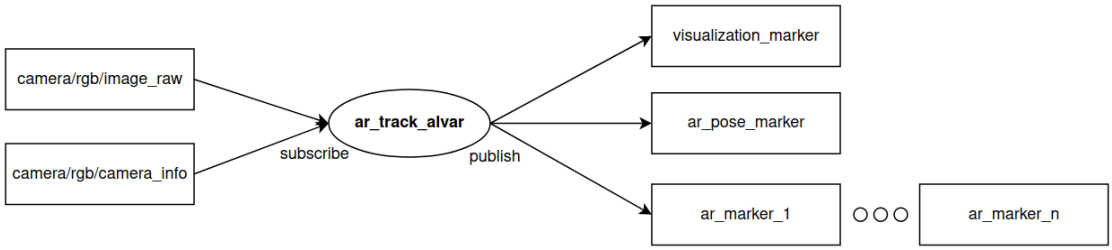


Figure 5.7: Graph of the ROS process regarding the `ar_track_alvar`.

Once a marker is detected, the package `ar_pkg` subscribes to the transform from the camera to the marker, and temporarily stores it to retrieve the coordinates later. This is necessary because during the exploration, the map is being generated, so at the moment when the marker is detected, the transform does not represent the position in the optimized frame. Therefore, the package `ar_sub` subscribes to the sent transform when the map optimization process is done, and writes it to file. Lastly, to determine the performance of the visual SLAM method, the ground truth can be observed in the RViz tool [31]. RViz is a toolkit that enables visualizing real-domain data generated by arbritary data structures and algorithms. Note that this ground truth is a constructed ground truth, as obtained by the fiducial approach, which is a simplification of the ground truth of the environment that was not available. This `ground_truth` topic is then published, which in turn the `ar_diff` package is subscribed to. This package then compares

the `ground_truth` with the observed marker location which can be read from file. This package calculates the Euclidean distance between the ground truth and observed marker position, and also is responsible for showing the ground truth and observed markers in the rviz tool for visualization purposes. This procedure is visualized in Figure 5.8.

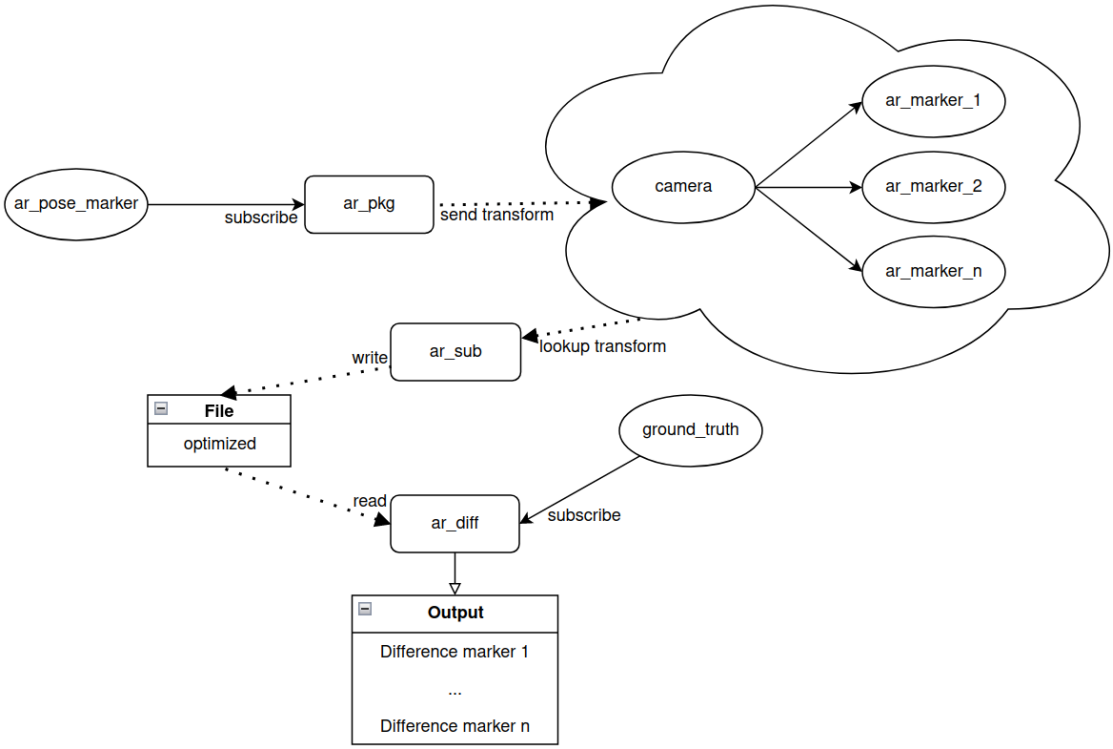


Figure 5.8: The evaluation procedure.

5.5 Summary

In this chapter, the implementation details of this study are presented. The robot Spot with its provided payload is tasked to walk around the environment of the Experiment Room to construct a map of the current environment by means of using the visual SLAM approach RTAB-Map. RTAB-Map also has a Feature Detection step in which the visual features used for the SLAM approach are obtained. In this study, a comparative analysis will be made of the performance of the constructed map using various visual features. Furthermore, an additional currently not available is introduced and will be integrated in the framework as well. Lastly, as the environment does not provide a ground truth, an evaluation method based on the fiducial principle is introduced. There approaches will be evaluated in chapter 7.

CHAPTER 6

Experiments

In this chapter, the conducted experiments will be discussed. The experiments focus not only on the currently available feature detection algorithms and how they relate to each other in various conditions and different datasets, but also in what way they can be separated. Also an additional feature detector is tried out.

6.1 Perspective

The first experiment being conducted is to gain an understanding of the current landscape and put the currently available feature detectors introduced in subsection 5.3.2 into perspective. However, as for the binary features, such as ORB and FREAK, they are very similar to BRIEF in terms of processing time, memory and localization performance, so only the BRIEF detector is included in the experiment [39].

In this experiment, the robustness of the visual feature algorithms are put to the test, with illumination variations in indoor environments as main objective. To achieve this, the dataset of an environment is captured during different lightning conditions in an apartment using an ASUS Zenfone AR phone with a running RTAB-Map Tango version [39]. Google Tango is a 3D augmented reality platform, which specifically brings augmented reality experiences to mobile devices, like smartphones and tablets [18].

During the creation of the multi-session map, the illumination conditions should be similar enough to the previous mapping session to a certain degree so that the map is able to follow the original trajectory, but it should experience sufficient lightning changes to realize the addition of new locations [39].

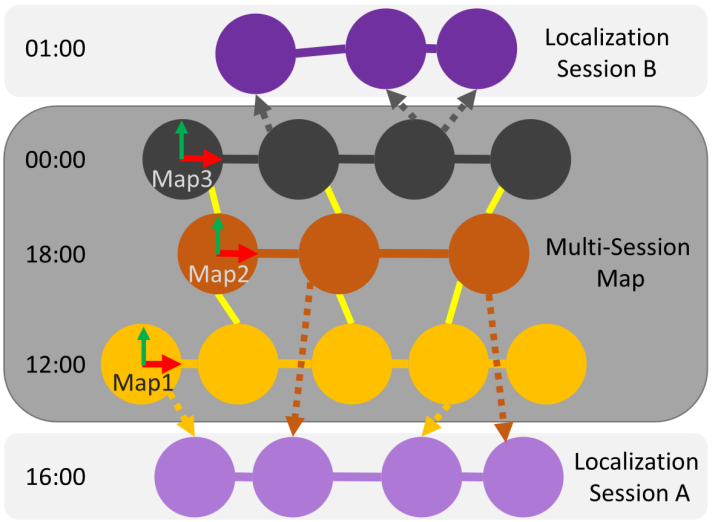


Figure 6.1: A visualization of the multi-session map. There are three sessions merged together and utilized to re-localize the during the localization sessions. Courtesy from [39].

To this extent, the structure of the map is represented as a pose-graph where the nodes represent each acquired image, and the links represent the six degrees of freedom transformations between them. This structure of the map is visualized in Figure 6.1. In this example, three sessions are taken during different times of the day, which are merged together in the SLAM phase by finding loop closures between the sessions. These loop closures are visualized as the yellow links. The two additional sessions are localization sessions, that represent two examples of experiment conduction in the localization phase. The corresponding frame of the re-localization is visualized with the dotted yellow link. The goal of the experiment is to gain an understanding of the robustness of the SLAM process using the various feature detection algorithms by re-localizing in the same environment during different times of the day [39].

6.2 Validation

Once the baseline is established, the gained insights to the various feature detection algorithms can be validated in a different environment, which could pose other challenges. One crucial aspect of the experiments is the dataset being used. While there are a variety of public dataset published for evaluating SLAM approaches, such as KITTI [1], TUM RGBD [2], EuRoC [3], for this study they are not appropriate. Among other issues, such that the KITTI dataset is specifically for autonomous driving and EuRoC is captured using a drone, they lack the placing of the AR tags that are being used for the evaluation part. Therefore, it was necessary to obtain the dataset using the provided hardware introduced in chapter 5.

The environment of the dataset is the Experiment Room of the Robot Learning Group of the University of Osaka. Generally, the purpose of this room is to conduct experiments, however, since the Robot Learning Group mostly focuses its research on the possibilities of home application robots, the setup of the room is similar to an apartment, including a kitchen and living area. Therefore, the environment is not as similar to convenience store, but nevertheless the performance could to a certain degree generalize to the convenience store as well. Furthermore, all the robots of the laboratory are stored in the room, so there are not multiple routes or trajectories for the robots due to the space. An example of the situation is shown in Figure 6.2.

[1] https://www.cvlibs.net/datasets/kitti/
[2] https://cvg.cit.tum.de/data/datasets/rgbd-dataset
[3] https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets

Figure 6.2: An example of the situation of the Experiment Room of the Robot Learning Group.

Before the dataset of the environment could be captured, some experiments were carried out to explore the environment. More specifically, the workings of the loop closure and the construction of a 3d overview of the environment were tested. As for the validation dataset, because the environment is not too large, either 2 repetitions of the same placing should suffice. Ideally, the dataset contains three marker placements, such that a discrepancy can be filtered out rather easily. However, during data capturing the robot Spot was not able to obtain all desired datasets, since it crashed down completely due to a misalignment on its arm, meaning that the robot Spot had to be sent back to the repair unit of Boston Dynamics, and was unfortunately not repaired in time to be used for this study.

Considering the placement of the markers itself, it is a requirement to think about the markers that they should be in view of the camera on the robot. So this means that the dataset should be tailored to the specific situation and setup of the experiment. Furthermore, ideally there would be a distinction between markers on easily detectable locations such as walls or more challenging places such as on corners, since they are both of value, such that the performance of the feature detection algorithms can be separated well.

The goal of this experiment is to research whether the established landscape of localization using various feature detection algorithms also holds up in the context of visual SLAM in another environment, and record a dataset that is able to create a separation between the performance of the various feature detectors.

6.3 Extension

The last experiment extends the previously discussed experiment by introducing an additional feature detection algorithm, and place it in the landscape of feature detectors discussed earlier. Therefore, the same obtained dataset can be used but the focus will be on the additional feature detector R2D2.

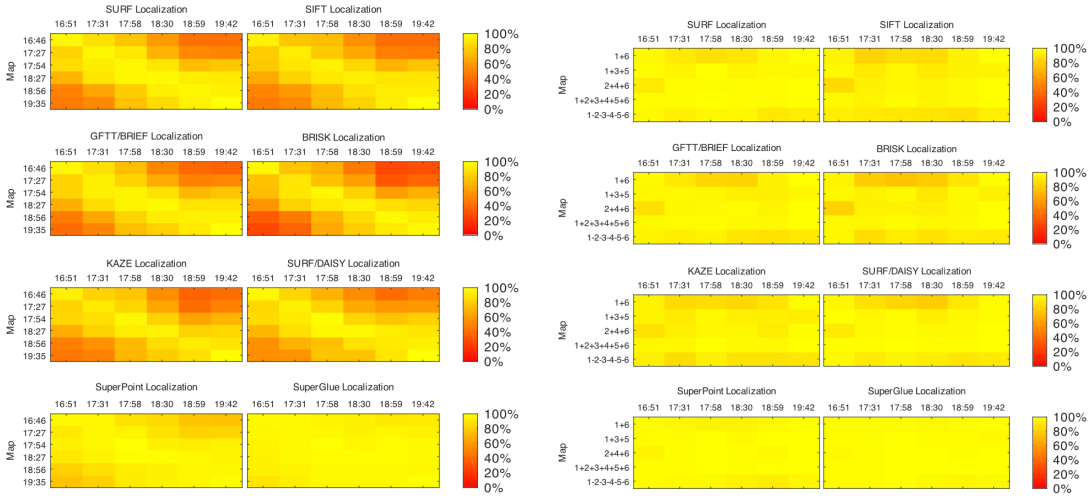
CHAPTER 7

Results

In this section the results of the previously discussed experiments will be presented. In total, there were three experiments, that give an overview of the performance of the currently within RTAB-Map available visual features.

7.1 Perspective

In the first experiment, the robustness of the various visual features were tested by determining the re-localization performance for a single mapping session. The percentage of re-localized frames can be visualized in Figure 7.1a. Note that the diagonals show the highest re-localizations performance, which is as expected since the illumination conditions of the re-localization mapping are then in line with those of the mapping session. Contrarily, the re-localization percentage is the lowest during day-time if the map is taken during night-time, or the other way around. This is shown by having more orange colors in the table on the non-diagonal. This is the case for all visual features, except for the SuperPoint, which shows to be the least dependent on whether the time of the mapping is in correspondence with the obtained localization map, both with and without SuperGlue.



(a) Single-session re-localization.

(b) Multi-session re-localization.

Figure 7.1: Localization percentage over six mapping sessions. Courtesy from [39].

Additionally, the same experiments were carried out using multi-session maps that are created using merging maps that are captured during different times of the day, and hence providing a

different illumination condition. The shown combinations of maps in Figure 7.1b are a combination of the two mapping sessions with the highest illumination variations (1+6), with a one-hour interval (1+3+5 and 2+4+6), and a thirty-minute interval (1+2+3+4+5+6). The merging of these maps have taken place without graph reduction, meaning that all nodes of all sessions are in tact, except for the 1-2-3-4-5-6 map, which represents the assembled maps with graph reduction enabled. The results show that except for SuperPoint, either with or without SuperGlue, all visual features have an increase in re-localization as more sessions are merged, meaning that the 1+2+3+4+5+6 map performs best with the 1-2-3-4-5-6 map not far behind.

7.2 Validation

In the validation experiments, a new dataset is obtained by exploring a new environment and gaining insights in how well the previous findings translate to this new environment.

7.2.1 Exploration

To this extent, the exploration experiments of the new dataset are visualized in Figure 7.2, which shows the Experiment Room of the Robot Learning Group. This viewer, the quality of the assembled points could be better, but it gives a good indication of the environment, with the kitchen section shown in the top and the living room section with a television and sofa shown in the middle section. Furthermore, we can see that not all sections are captured enough with the camera to properly show the situation. For example, the left and bottom walls are lined with stand-by robots and working places, but those are not visualized correctly, since the hallways are very narrow, which prevents the robot from turning and capturing the situation after turning.

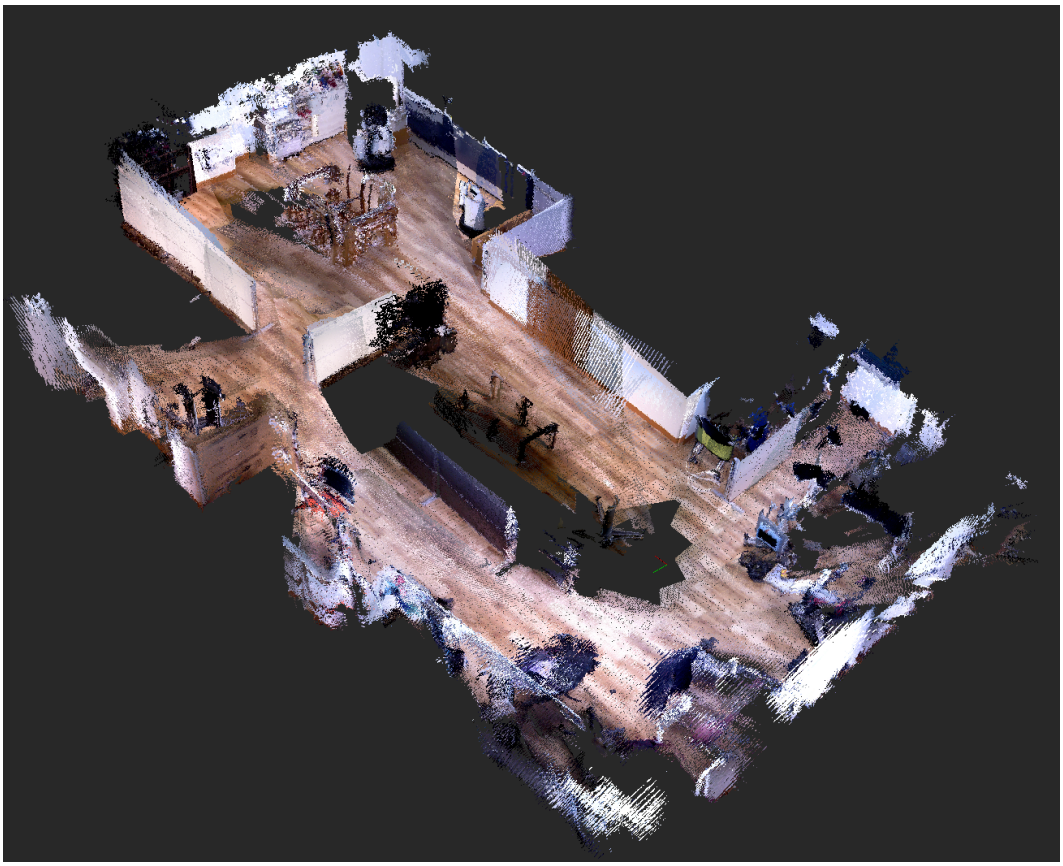
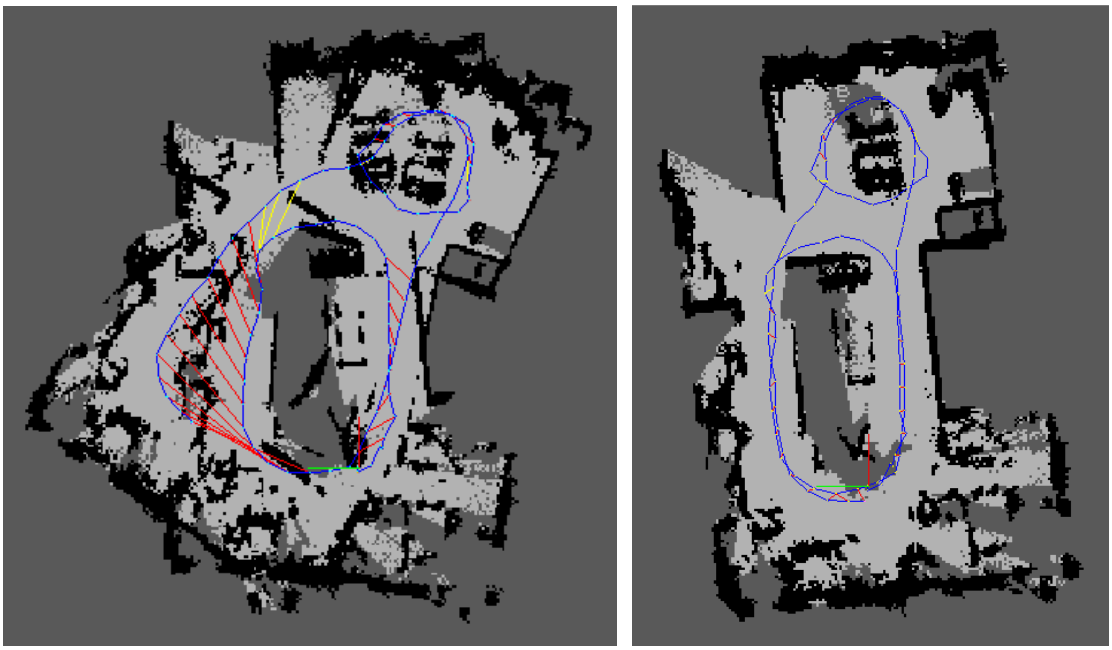


Figure 7.2: 3D pointcloud viewer of a single trajectory through the environment.

This 3D point cloud visualization is obtained by using both the lidar sensor and the camera

introduced in section 5.2. This combination results in the fact that the lidar sensor can be precise in the demarcation of the walls and the camera can detect loop closures to correct the accumulated drift. The effect of loop closure is visualized in Figure 7.3, where the trajectory of the robot is shown as the blue line. On the left image, i.e. before loop closure, the entire environment is slightly rotated, but since the robot is able to recognize the place it has been before, it is able to adjust the accumulated drift and correct its trajectory. This discrepancy between the first and second time the robot visits a place in the environment is shown in red lines. After the correction, Figure 7.3b shows the environment with straight walls, without being rotated.



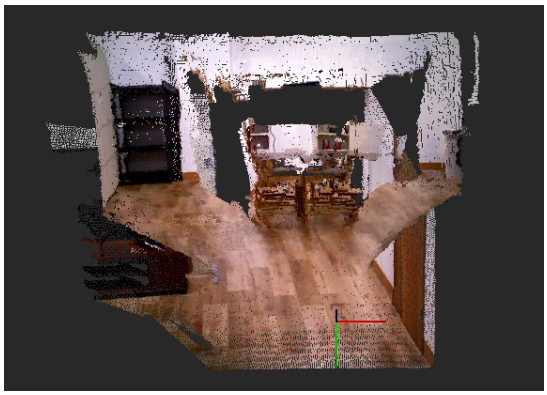
(a) Before (b) After

Figure 7.3: The obtained map before and after applying loop closure.

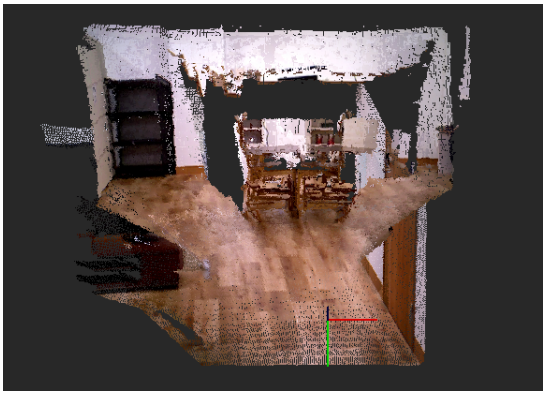
The exploration experiments entailed showing the constructed maps to get an idea of the environment.

7.2.2 Qualitative results

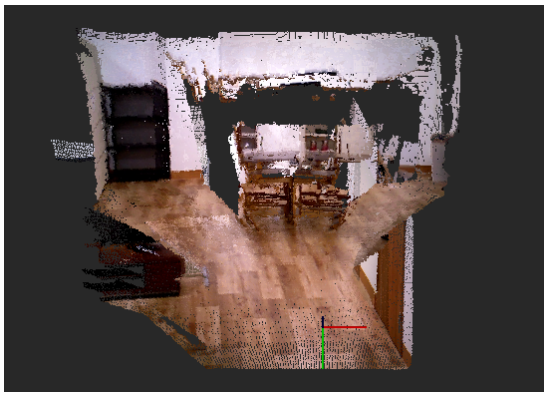
Previous experiments have shown that re-localization in an environment using RTAB-Map performs very similarly, regardless of the visual feature being used, with SuperPoint being the exception. Therefore, to create a separation between the visual features, another experiment is conducted. In this section, the qualitative results of a run are shown in which the trajectory is only a straight line and is through the hallway with the living room on the left side and the wall on the right side. The trajectory ends before the dinner table. In Figure 7.4, we see that there is a distinction between the SuperPoint visual features on the one hand, and the other visual features on the other hand. Visually speaking, we can assess that SuperPoint performs the worst among all the feature detectors. Note that while SuperPoint's visualization appears to have a cracked pattern, this is actually due to the fact that it is more zoomed in, because SuperPoint was not able to match at the beginning of the trajectory. This is surprising, as in the re-localization experiment SuperPoint emerged as the winner. Furthermore, the differences between the other feature detectors are subtle but present. For example, the visualization obtained by KAZE is less precise seen by the fact that less of the floor is mapped. This is not the case for the other feature detectors.



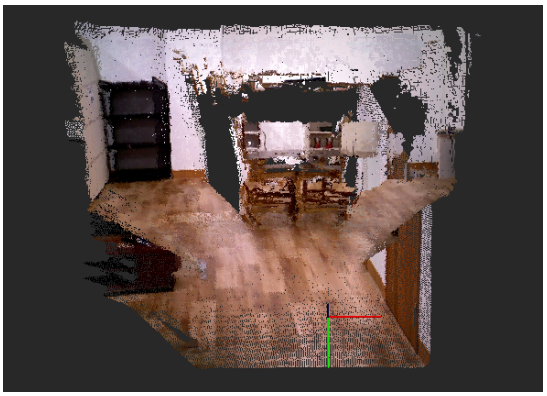
(a) SURF



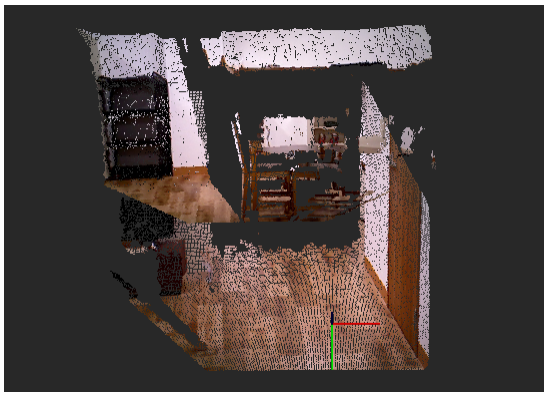
(b) SIFT



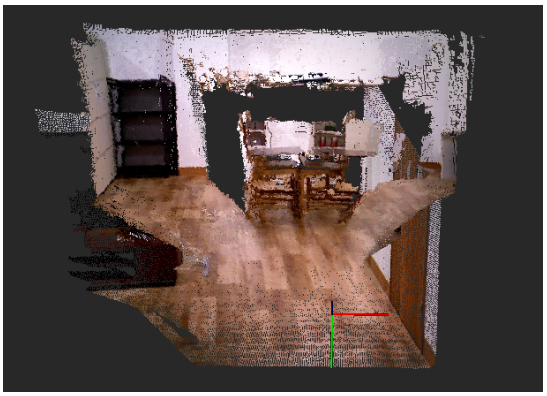
(c) GFTT/BRIEF



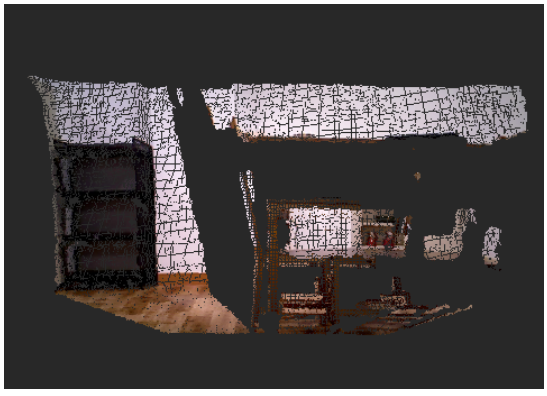
(d) BRISK



(e) KAZE



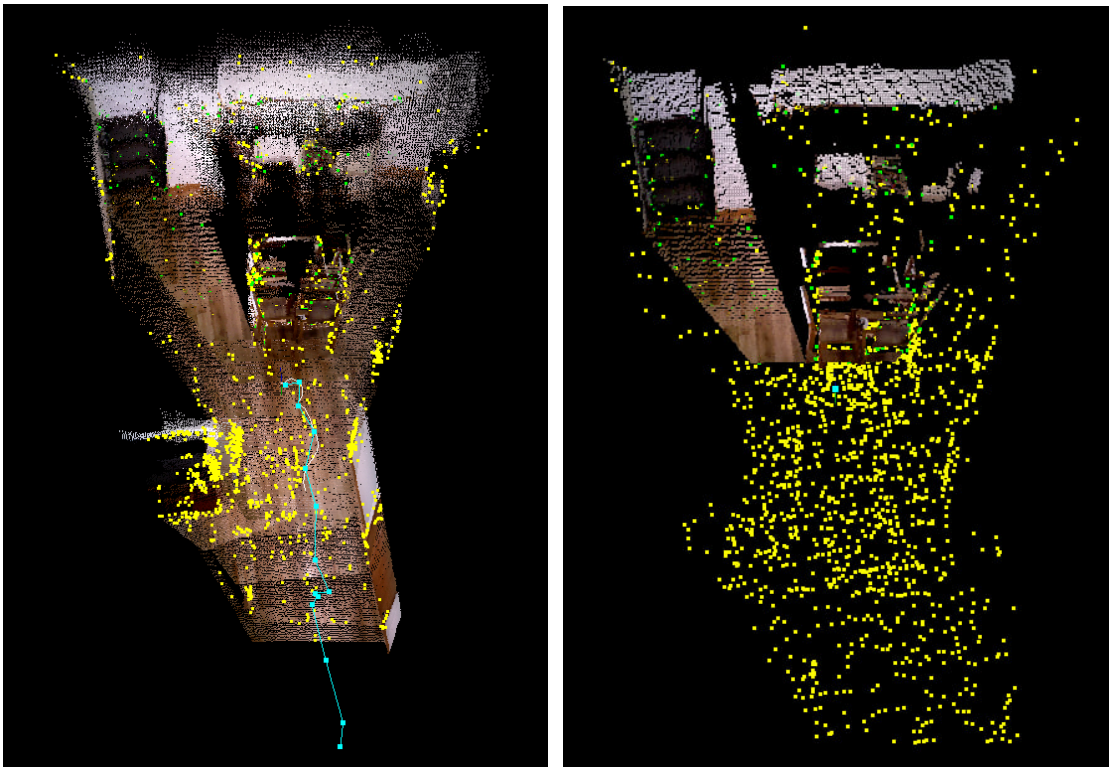
(f) SURF/DAISY



(g) SuperPoint

Figure 7.4: The qualitative results of a single trajectory experiment.

In this specific case, SuperPoint seems unable to match the first couple of frames, meaning that the entire hallway is not correctly matched, and only is able to map its current location just before the dinner table at the end of the captured data. Upon further inspection, the construction of the map visualized in Figure 7.5 shows that SuperPoint is in fact able to find keypoints, but not able to match them. Every dot is a keypoint, of which the yellow color indicates that it is a new keypoint but not unique, whereas a green color indicates that it is a new keypoint and also unique. Therefore, the fact that the SuperPoint construction and SURF construction both have a yellow dots, in combination with the fact that the SuperPoint construction shows yellow dots, which are not unique, and is still not able to match them to construct the environment indicates that it struggles in this particular situation. This could be due to the fact that SuperPoint is a deep-learning based model, meaning that it is dependent on the data it is trained, and maybe in this particular case could not be generalized well to this environment.



(a) SURF (b) SuperPoint

Figure 7.5: Comparison between the construction of the map using SURF and SuperPoint.

7.2.3 Quantitative results

The quantitative results are examined in this subsection. Note that because of the Spot robot requiring repairment in Boston, these experiments were performed with a substandard dataset. Nonetheless, the findings can gain some insight in the performance of the feature detectors under challenging conditions. Table 7.1 shows the results, which presents the mean and standard variance of every marker after the visual SLAM process using every feature detector over three runs. Since these reported values are errors, the lower the value is, the better. Hence, the error presented in bold is the best performing feature detector for every marker. Interestingly, the three AR markers each have a different best-performing feature detector, namely BRISK for AR marker 1, SURF/DAISY for AR marker 2 and SuperPoint for AR marker 3. Notice how the values differ significantly, meaning that probably AR marker 1 was more difficulty placed as opposed to AR marker 3. This actually makes sense, since marker 1 was placed along the wall where the Spot robot did not have enough room to turn to the side, whereas marker 3 was placed in the middle of the room, where it was visible from multiple sides. This shows

that an experiment where this quantitative evaluation approach is used has to be tailored to the environment. Unfortunately, as discussed in the previous subsection, the SuperPoint was not able to map the beginning of the trajectory, meaning that the first two markers were not captured and could therefore not be included in the results table. This was the case for all three runs. However, for the marker it was able to map, the error value is the lowest of all feature detectors, which is promising for better conditions. Among the feature detectors, it is interesting to see that there are considerable difference in their reliability. KAZE is, apart from the error value it obtained, very stable in its performance, wheras SIFT performs especially for marker 1 very unstable.

	AR marker 1	AR marker 2	AR marker 3
SURF	2.416 ± 0.538	0.107 ± 0.018	0.268 ± 0.322
SIFT	2.535 ± 1.001	0.107 ± 0.015	0.311 ± 0.296
GFTT/BRIEF	2.779 ± 0.482	0.075 ± 0.028	0.081 ± 0.026
BRISK	**2.389 ± 0.515**	0.106 ± 0.018	0.071 ± 0.035
KAZE	3.020 ± 0.088	0.153 ± 0.032	0.066 ± 0.004
SURF/DAISY	3.052 ± 0.129	**0.099 ± 0.027**	0.076 ± 0.019
SuperPoint	-	-	**0.054 ± 0.022**

Table 7.1: The quantitative results given with the mean and standard deviation over three runs. The lower the better. The best feature detector for every marker is printed in bold.

7.3 Extension

In this section, the results of the additional feature detection algorithm incorporated in RTAB-Map will be presented. Specifically, the feature detector R2D2 was incorporated into RTAB-Map, but unfortunately, RTAB-Map was not able to construct a proper map of the environment using the by R2D2 provided keypoints and descriptors due to losing visual odometry. When visual odometry is lost, this means that there were not enough keypoints found and matched between the frames, meaning that the constructed map is empty. Upon further inspection, the keypoints were correctly found, as shown in Figure 7.6, where there are features found along a certain line or interesting keypoints such as edges. Furthermore, the size of the circle shows the confidence and we see that there are cases in which a large circle is placed over a marker. However, we also see some dubious keypoints found, for example in the frame 4, 7 and 8, keypoints are found along the floor, which should not be a repeatable and reliable keypoint. Therefore, it was not possible to correctly integrate the R2D2 feature detector in the RTAB-Map visual SLAM approach.

Figure 7.6: The found keypoints using the R2D2 feature detector. Note that the larger the circle is, the more confident the detector is.

7.4 Conclusion

In this chapter, the results of the in chapter 6 introduced experiments were presented. This included the perspective experiment, which showed promising re-localization performance in both single-session and multi-session mapping. Interestingly, while SuperPoint outperformed all other visual features in the re-localization experiments, the opposite was true for the additional validation experiments. This may have something to do with the fact that the data set was substandard, but it may also be an issue of deep-learning based visual features in combination with RTAB-Map. This could also explain the fact that the integration of R2D2 did not succeed in constructing a map. Nonetheless, the experiments have also resulted in an exploration map of the used environment, and have demonstrated the effect of loop closure detection.

CHAPTER 8

Conclusions

This study set out to gain insights in the applicability and performance of visual features in a visual SLAM approach. While lidar SLAM has been discussed and researched to a great extent, visual SLAM in much less degree. Therefore, in this study a comparative analysis is made between the feature detectors and various visual SLAM approaches. Furthermore, the re-localization performance of these visual features is highlighted together with further experiments using these visual features in an unknown environment of the Experiment Room. To assess the obtained map of the environment using the diverse visual features an evaluation approach is proposed. The concluding findings will be examined in this chapter.

8.1 Discussion

First of all, the captured datasets were not in the desired quality. This was the result of the robot Spot not being in optimal condition, meaning that the robot often collapsed which made a lot of datasets unusable. Eventually, the robot Spot had to be send back to Boston Dynamics for reparation, but unfortunately did not make it back to the Robot Learning Group in time to be used for this study. While the datasets could also be captured using a wheeled-robot, due to the applicability of a legged-robot, the choice was made to experiment with the substandard datasets. This could still give an indication of the performance of the feature detectors and visual SLAM approach, but it is not possible to make generalizations of the performance. Hence, this study only answers the research questions in the specific case of the Experiment Room being the environment. Similarly, the fact that no repeating experiments were possible, to validate the results of this study, more thorough testing has to be done.

8.2 Future work

The results indicate that the combination of lidar and visual SLAM approach performed considerably well, with the layout of the Experiment Room being quite distinctive. However, along the walls of the environment and in the center, it seems that it was not captured enough by the camera. Future research might explore the possibility of improving the quality of regions that are brought less into view. Additionally, the deep-learning based feature detectors did not perform as expected at all. This has to be investigated further to incorporate deep-learning based feature detectors better into the RTAB-Map framework. One direction of research could be the effect of the pre-training step and data specifically on the performance of the visual features to improve the current mapping using visual SLAM.

8.3 Conclusion

The research questions of this study are:

RQ1. What are the currently available feature detectors for visual SLAM methods, and how do they compare to each other?

RQ2. Is it possible to integrate a new feature detector, namely R2D2, into RTAB-Map, and how does its performance compare to the existing feature detectors?

RQ3. How can map assessment be performed of a map produced by visual SLAM when ground truth is not available?

To formulate an answer on the first question, the currently available feature detectors were categorized in three categories, namely binary features, floating-point features and deep-learning based features, and every category was theoretically discussed, from which the conclusion was that there is no universal and optimal approach, as the best choice is generally dependent on the specific task, the setup and the existing distortions in the available data. The corresponding experiments showed that SuperPoint was a special case, where it was the complete winner in the re-localization experiments, but performs by far the worst in the experiment in the Experiment Room. This is in line with the hypothesis that the performance of the feature detectors is to a certain extent dependent on external factors. Nonetheless, the reason that SuperPoint did perform this unexpectedly bad is an interesting direction for future research. To this extent, SuperPoint may have been experiencing the same issues as R2D2, also a deep-learning based approach, which also formulates an answer to the second question. While it was possible to integrate R2D2 into the RTAB-Map framework as the used visual features, its performance could not be determinant as it was not possible to obtained a map by means of the RTAB-Map visual SLAM approach. Whether this is the same cause as SuperPoint, can be investigated as future work. The other feature detectors were compared in the both qualitative and quantitative manner. Whereas the qualitative manner was restricted to looking which map is more visually appealing, the quantitative map was able to associate the performance to a value. In these experiments, as there was no ground truth available, the map assessment of a map produces by RTAB-Map could be performed using an evaluation approach based on the fiducial principle, where fiducials are placed in the environment that can be compared to the location of the observed fiducial. Therefore, this answers the third research question, however, more thorough testing has to be done to evaluate the exact performance and applicability of this evaluation approach. To conclude, this thesis has provided a deeper insight into the feature detection step of the visual SLAM approach RTAB-Map specifically in the combination with the robot Spot and the ROS framework. While this study has answered the research questions, the performance of the R2D2 feature detector and more robust findings on the evaluation procedure still remains to be seen in future work.

Bibliography

[1] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. Kaze features. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*, pages 214–227. Springer, 2012.

[2] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the ekf-slam algorithm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568. IEEE, 2006.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[4] Amanda Bouman, Muhammad Fadhil Ginting, Nikhilesh Alatur, Matteo Palieri, David D Fan, Thomas Touma, Torkom Pailevanian, Sung-Kyun Kim, Kyohei Otsu, Joel Burdick, et al. Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2518–2525. IEEE, 2020.

[5] Alan Bundy and Lincoln Wallen. Difference of gaussians. *Catalogue of Artificial Intelligence Tools*, pages 30–30, 1984.

[6] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, pages 778–792. Springer, 2010.

[7] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[8] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[9] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[10] Kaichang Di, Wenhui Wan, Hongying Zhao, Zhaoqin Liu, Runzhi Wang, and Feizhou Zhang. Progress and applications of visual slam. *Journal of Geodesy and Geoinformation Science*, 2(2):38, 2019.

[11] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[12] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. *arXiv preprint arXiv:1905.03561*, 2019.

[13] Daniel Dworakowski, Christopher Thompson, Michael Pham-Hung, and Goldie Nejat. A robot architecture using contextslam to find products in unknown crowded retail environments. *Robotics*, 10(4):110, 2021.

[14] István Faragó. A modified iterated operator splitting method. *Applied mathematical modelling*, 32(8):1542–1551, 2008.

[15] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596. IEEE, 2005.

[16] Maksim Filipenko and Ilya Afanasyev. Comparison of various slam systems for mobile robot in an indoor environment. In *2018 International Conference on Intelligent Systems (IS)*, pages 400–407. IEEE, 2018.

[17] Wolfgang Förstner and Eberhard Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*, volume 6, pages 281–305. Interlaken, 1987.

[18] Mark Froehlich, Salman Azhar, and Matthew Vanture. An investigation of google tango® tablet for low cost 3d scanning. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 34. IAARC Publications, 2017.

[19] Constantine Gidaris. The rise of the robots: Technocapitalism and the policing of race. *Fast Capitalism*, 18(1), 2021.

[20] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.

[21] H-M Gross, H Boehme, Ch Schroeter, Steffen Müller, Alexander König, Erik Einhorn, Ch Martin, Matthias Merten, and Andreas Bley. Toomas: interactive shopping guide robots in everyday use-final implementation and experiences from long-term field trials. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2005–2012. IEEE, 2009.

[22] Robin P Guan, Branko Ristic, and Liuping Wang. Combining kld-sampling with gmapping proposal for grid-based monte carlo localization of a moving robot. In *2017 20th International Conference on Information Fusion (Fusion)*, pages 1–8. IEEE, 2017.

[23] Erico Guizzo. By leaps and bounds: An exclusive look at how boston dynamics is redefining robot agility. *IEEE Spectrum*, 56(12):34–39, 2019.

[24] Dirk Hahnel, Dirk Schulz, and Wolfram Burgard. Map building with mobile robots in populated environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 496–501. IEEE, 2002.

[25] Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[26] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[27] Mahmoud Hassaballah, Hammam A Alshazly, and Abdelmgeid A Ali. Analysis and evaluation of keypoint descriptors for image matching. *Recent Advances in Computer Vision: Theories and Applications*, pages 113–140, 2019.

[28] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. Comparative evaluation of binary features. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II 12*, pages 759–773. Springer Berlin Heidelberg, 2012.

[29] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016.

[30] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, Anand Ambrosi, Camden Whitehead, and Nikolaos Vitzilaios. Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers. *Journal of Intelligent & Robotic Systems*, 101:1–26, 2021.

[31] Hyeong Ryeol Kam, Sung-Ho Lee, Taejung Park, and Chang-Hun Kim. Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, 60:337–345, 2015.

[32] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Uwe Klingauf, and Oskar von Stryk. Hector open source modules for autonomous mapping and navigation with rescue robots. In *Robot Soccer World Cup*, pages 624–631. Springer, 2013.

[33] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *2011 IEEE international symposium on safety, security, and rescue robotics*, pages 155–160. IEEE, 2011.

[34] Kurt Konolige, Giorgio Grisetti, Rainer Kümmerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent. Efficient sparse pose adjustment for 2d mapping. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 22–29. IEEE, 2010.

[35] Brett Koonce and Brett Koonce. Vgg network. *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*, pages 35–50, 2021.

[36] Mathieu Labbe and Francois Michaud. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, 29(3):734–745, 2013.

[37] Mathieu Labbe and François Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2661–2666. IEEE, 2014.

[38] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[39] Mathieu LabbÃ© and FranÃ§ois Michaud. Multi-session visual slam for illumination-invariant re-localization in indoor environments. *Frontiers in Robotics and AI*, 9, 2022.

[40] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. Ieee, 2011.

[41] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[42] Jiayi Ma, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. Image matching from handcrafted to deep features: A survey. *International Journal of Computer Vision*, 129:23–79, 2021.

[43] Steve Macenski and Ivona Jambrecic. Slam toolbox: Slam for the dynamic world. *Journal of Open Source Software*, 6(61):2783, 2021.

[44] Elmar Mair, Gregory D Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II 11*, pages 183–196. Springer, 2010.

[45] Chelsia Rose Maricus. See 'spot' save: Robot dogs join the new york fire department. `https://www.nytimes.com/2022/03/17/nyregion/fdny-boston-dynamics-spot-robot.html`. (accessed: 03.11.2022).

[46] Michael Montemerlo and Sebastian Thrun. Fastslam 1.0. *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, pages 27–62, 2007.

[47] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[48] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee, 2004.

[49] Ministry of Internal Affairs and Japan. Communications. Population census,tabulation on internal migration for population; 2015. `https://www.stat.go.jp/english/data/kokusei/2015/summary.html`. (accessed: 10.11.2022).

[50] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[51] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2d2: repeatable and reliable detector and descriptor. *arXiv preprint arXiv:1906.06195*, 2019.

[52] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, pages 430–443. Springer, 2006.

[53] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[54] Artur Sagitov, Ksenia Shabalina, Leysan Sabirova, Hongbing Li, and Evgeni Magid. Artag, apriltag and caltag fiducial marker systems: Comparison in a presence of partial marker occlusion and rotation. In *ICINCO (2)*, pages 182–191, 2017.

[55] P Sankalprajan, Thrilochan Sharma, Hamsa Datta Perur, and Prithvi Sekhar Pagala. Comparative analysis of ros based 2d and 3d slam algorithms for autonomous ground vehicles. In *2020 International Conference for Emerging Technology (INCET)*, pages 1–6. IEEE, 2020.

[56] Sören Schwertfeger, Adam Jacoff, Johannes Pellenz, and Andreas Birk. Using a fiducial map metric for assessing map quality in the context of robocup rescue. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 208–214. IEEE, 2011.

[57] Xuelun Shen, Cheng Wang, Xin Li, Yifan Peng, Zijian He, Chenglu Wen, and Ming Cheng. Learning scale awareness in keypoint extraction and description. *Pattern Recognition*, 121:108221, 2022.

[58] Sawako Shirahase. *Social Stratification in an Aging Society with Low Fertility: The Case of Japan*. Springer Nature, 2022.

[59] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[60] Sivic and Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings ninth IEEE international conference on computer vision*, pages 1470–1477. IEEE, 2003.

[61] S Thrun, W Burgard, and D Fox. Probabilistic robotics. ma, 2005.

[62] Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.

[63] Yurun Tian, Bin Fan, and Fuchao Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 661–669, 2017.

[64] Ali Tourani, Hriday Bavle, Jose Luis Sanchez-Lopez, and Holger Voos. Visual slam: What are the current trends and what to expect? *arXiv preprint arXiv:2210.10491*, 2022.

[65] Daniel Troniak, Junaed Sattar, Ankur Gupta, James J. Little, Wesley Chan, Ergun Calisgan, Elizabeth Croft, and Machiel Van der Loos. Charlie rides the elevator – integrating vision, navigation and manipulation towards multi-floor robot locomotion. In *2013 International Conference on Computer and Robot Vision*, pages 1–8, 2013.

[66] Kazuyoshi Wada. New robot technology challenge for convenience store. In *2017 IEEE/SICE International Symposium on System Integration (SII)*, pages 1086–1091. IEEE, 2017.

[67] Joachim Weickert. A review of nonlinear diffusion filtering. In *International conference on scale-space theories in computer vision*, pages 1–28. Springer, 1997.

[68] Felix H Wullschleger, Kai O Arras, and Sjur J Vestli. A flexible exploration framework for map building. In *1999 Third European Workshop on Advanced Mobile Robots (Eurobot'99). Proceedings (Cat. No. 99EX355)*, pages 49–56. IEEE, 1999.

[69] Po-Yu Yang, Tzu-Hsuan Chang, Yu-Hao Chang, and Bing-Fei Wu. Intelligent mobile robot controller design for hotel room service with deep learning arm-based elevator manipulator. In *2018 International Conference on System Science and Engineering (ICSSE)*, pages 1–6, 2018.

[70] Azalea Yunus and Stacy A Doore. Responsible use of agile robots in public spaces. In *2021 IEEE International Symposium on Ethics in Engineering, Science and Technology (ETHICS)*, pages 1–5. IEEE, 2021.